- 合理、完善的知识体系结构
- 内容丰富，重点突出，应用性强
- 免费提供相关程序源代码下载
- 深入、详细剖析 MATLAB 工程应用技术



MATLAB 数字图像处理

网上提供源代码下载
www.cmpbook.com

张德丰　等编著

机械工业出版社
CHINA MACHINE PRESS

ISBN 978-7-111-25735-6
策　划：丁 诚　吴鸣飞
封面设计：子时文化 ZiShi Culture

本书利用MATLAB图像处理工具箱进行数字图像处理的设计与应用，紧密联系实际，以具体的实例介绍了函数的使用方法。在实例中强调了如何用MATLAB图像处理工具箱解决图像处理中的问题、难题，节省了图像处理的时间，提高了图像处理的效率。

本书详细介绍了数字图像处理技术及利用MATLAB进行图像处理的方法和技巧，强调了图像处理的理论和应用相结合的方法，并给出了大量数字图像处理技术的MATLAB实现程序。

本书可作为高等理工科院校电子信息、通信工程、信号与信息处理学科的本科生教材，也可作为研究生以及从事图像研究的科研工作者的学习参考用书。

上架建议 计算机/辅助设计



编辑热线：(010) 88379753　88379739

地址：北京市百万庄大街22号　邮政编码：100037
联系电话：(010)68326294　网　址：http://www.cmpbook.com(机工门户网)
(010)68993821　E-mail:cmp@cmpbook.com
购书热线：(010)88379639　(010)88379641　(010)88379643

定价：39.00元

MATLAB 工程应用书库

MATLAB 数字图像处理

张德丰　等编著



机 械 工 业 出 版 社

本书利用 MATLAB 图像处理工具箱进行数字图像处理的设计与应用，简洁明了地指出了所介绍的函数与方法的理论背景，同时又紧密联系实际应用，以具体的实例说明了函数的使用方法。在实例中强调了如何用 MATLAB 图像处理工具箱解决图像处理中的问题、难题，节省了图像处理的时间，提高了图像处理的效率。

本书详细介绍了数字图像处理技术及利用 MATLAB 进行图像处理的方法和技巧，强调了图像处理的理论和应用相结合的方法，并给出了大量数字图像处理技术的 MATLAB 实现程序。

本书可作为高等理工科院校电子信息、通信工程、信号与信息处理学科的本科生教材，也可作为研究生以及从事图像研究的科研工作者的学习参考用书。

图书在版编目（CIP）数据

MATLAB 数字图像处理 / 张德丰等编著. —北京：机械工业出版社，2009.1
（MATLAB 工程应用书库）
ISBN 978-7-111-25735-6

Ⅰ. M… Ⅱ. 张… Ⅲ. 图像处理－计算机辅助计算－软件包，MATLAB Ⅳ. TP391.41

中国版本图书馆 CIP 数据核字（2008）第 192914 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
策划编辑：丁　诚　吴鸣飞
责任编辑：丁　诚　吴超莉
责任印制：杨　曦

三河市国英印务有限公司印刷

2009 年 1 月第 1 版・第 1 次印刷
184mm×260mm・22 印张・540 千字
0001－4000 册
标准书号：ISBN 978-7-111-25735-6
定价：39.00 元

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话：（010）68326294　68993821
购书热线电话（010）88379639　88379641　88379643
编辑热线电话（010）88379753　88379739

前　言

MATLAB 的英文全称是 Matrix Laboratory（矩阵实验室），一开始它是一种专门用于矩阵数值计算的软件。从这一点上也可以看出，它在矩阵运算方面有自己的特点。实际上，MATLAB 中的绝大多数运算都是通过矩阵这一形式完成的。正是这一特点决定了 MATLAB 在处理数字图像上的独特优势。从理论上讲，图像是一种二维的连续函数，然而在计算机上对图像进行数字处理时，首先必须对其在空间和亮度上进行数字化，即图像的采样和量化过程。通过对二维图像的均匀采样，就可以得到一幅离散化成 $M \times N$ 样本的数字图像。该数字图像是一个整数阵列，因而用矩阵来描述是最直观和最简便的了。而 MATLAB 的长处就是处理矩阵运算，因此用 MATLAB 处理数字图像非常方便。

MATLAB 支持索引色图像、灰度图像、二值图像、RGB 图像和多帧图像阵列五种图像类型，支持 BMP、GIF、HDF、JPEG、PCX、PNG、TIFF、XWD、CUR、ICO 等图像文件格式的读、写和显示。MATLAB 对图像的处理功能主要集中在它的图像处理工具箱（Image Processing Toolbox）中。图像处理工具箱由一系列支持图像处理操作的函数组成，可以进行诸如几何操作、线性滤波和滤波器设计、图像变换、图像分析与图像增强、二值图像操作以及形态学处理等图像处理操作。

在数字图像处理领域对问题的求解通常需要宽泛的实验工作，包括软件模拟和大量样本图像的测试。虽然典型算法的开发是基于理论支持的，但这些算法的实现几乎总是要求参数估计，并常常进行算法修正与候选求解方案的比较。这样，灵活的、综合的以及由许多资料证明的软件开发环境就是一个关键因素。这些因素在开销、开发时间和图像处理求解方法上都具有重要意义。

尽管它很重要，但却很少有以教材形式编写的涉及数字图像处理原理和软件实现方面的文献。而本书恰好是以概要形式讲述基本理论并紧密结合实际应用研究为目的而编写的。

全书共分 11 章，第 1 章介绍图像处理与 MATLAB 2007a 简介；第 2 章介绍图像的编码和解码；第 3 章介绍图像复原；第 4 章介绍图像处理的相关操作；第 5 章介绍图像频域变换；第 6 章介绍图像处理中的代数运算及几何变换，并重点介绍图像几何的变换；第 7 章介绍图像增强；第 8 章介绍图像分割与边缘检测；第 9 章介绍小波分析及其在 MATLAB 中的应用，主要讲述小波技术在图像处理中的应用，并详细展开小波分析在图像增强中的应用，如基于小波的图像降噪和压缩、小波的融合技术、小波包在图像边缘检测中的应用；第 10 章介绍图像特征的描述；第 11 章介绍 MATLAB 图像处理的应用，主要介绍 MATLAB 在医学和遥感图像处理方面的应用。

本书结合了作者多年来的教学实践和研究经验，并力图体现以下三个特点。第一，将图像处理理论和应用举例相结合，系统介绍了数字图像处理技术的相关知识和内容，每介绍一个知识点的理论都有相应的应用举例，使读者对数字图像处理学科有一个全面的了解；第二，根据国内有关专业本科生和研究生的培养规划，介绍了图像处理技术方面的新理论、新技术、新标准和新应用，使读者充分了解图像处理技术的新发展和新应用；第三，注重

MATLAB 图像处理功能在实际生活中的应用举例，使读者学以致用。

本书力求内容丰富、图文并茂、文字流畅，将会成为一本学习和使用 MATLAB 数字图像处理方面有价值的参考书。

参加本书编写的有张德丰、许华兴、王旭宝、王孟群、邓恒奋、卢国伟、卢焕斌、伍志聪、庄文华、庄浩杰、许业成、何沛彬、何佩贤、张水兰、张坚、李勇杰、李秋兰、李美妍、陈运英、陈景棠、梁家科、黄达中、陈楚明、林健锋、梁劲强、林振满、周品。

由于时间仓促，加上作者水平所限，错误或疏漏之处在所难免，敬请读者批评指正。

编　者

目 录







第 1 章 图像处理与 MATLAB 2007a 简介

1.1 概述

1.1.1 MATLAB 概述

MATLAB 是一种面向科学与工程计算的高级语言，允许用数学形式的语言来编写程序，比 BASIC、FORTRAN 和 C 语言更加接近书写计算公式的思维方式，用 MATLAB 编写程序犹如在演算纸上排列出公式与求解问题一样。因此，MATLAB 语言也可以通俗地称为“演算纸”式科学算法语言。它编写简单，编程效率高，易学易懂。

MATLAB 诞生于 20 世纪 70 年代，它的编写者是 Cleve Moler 博士和他的同事。当时 Cleve Moler 博士和他的同事开发了 EISPACK 和 LINPACK 的 FORTRAN 子程序库，这两个程序主要是求解线性方程的程序库。但是 Cleve Moler 发现，学生使用这两个程序库时有困难，主要原因是接口程序不好写，很费时间。于是，Cleve Moler 自己动手，在业余时间里编写了 EISPACK 和 LINPACK 的接口程序，Cleve Moler 给这个接口程序取名为 MATLAB，意为矩阵（Matrix）和实验室（Laboratory）的组合。

1984 年，Cleve Moler 和 John Little 成立了 MathWorks 公司，正式把 MATLAB 推向市场，并继续进行 MATLAB 的开发。1993 年 MathWorks 公司发布了 MATLAB 4.0，1995 年发布了 MATLAB 4.2C 版（For Windows 3.x），1997 年发布了 MATLAB 5.0，2000 年 10 月发布了 MATLAB 6.0，2002 年 8 月发布了 MATLAB 6.5，2004 年 9 月发布了 MATLAB 7.0，2006 年 3 月发布了 MATLAB 2006a，2006 年 9 月发布了 MATLAB 2006b，2007 年 3 月发布了最新的 MATLAB 2007a。每一新版本的推出都使 MATLAB 有了很大的改进——界面越来越友好，内容越来越丰富，功能越来越强大，帮助系统越来越完善。MATLAB 2007a 增强了 2 个新产品模块，同时还升级和修正了 82 个产品模块。

MATLAB 擅长数值计算，能处理大量的数据，而且效率比较高。MathWorks 公司在此基础上加强了 MATLAB 的符号计算、文字处理、可视化建模和实时控制能力，增强了 MATLAB 的市场竞争力，使 MATLAB 成为市场上主流的数值计算软件。

MATLAB 产品族支持从概念设计、算法开发、建模仿真到实时实现的理想的集成环境。无论是进行科学研究还是产品开发，MATLAB 产品族都是必不可少的工具。MATLAB 产品族具有如下主要产品。

1）MATLAB：MathWorks 公司所有产品的数值分析和图形基础环境。MATLAB 将 2D

和 3D 图形、MATLAB 语言编程集成到一个单一的、易学易用的环境中。

2）MATLAB 工具箱：一系列专用的 MATLAB 函数库，用于解决特定领域的问题。工具箱是开放的、可扩展的，可以查看其中的算法或开发用户需要的算法。

3）MATLAB 编译器：将 MATLAB 语言编写的 M 文件自动转换成 C 或 C++文件，支持用户进行独立的应用开发。结合 MathWorks 公司提供的 C/C++数学库和图形库，用户可以利用 MATLAB 快速地开发出功能强大的独立应用系统。

4）Simulink：是结合了框图界面和交互仿真能力的非线性动态系统仿真工具。它以 MATLAB 的数学、图形和语言为基础。

5）Stateflow：与 Simulink 框图模型相结合，描述复杂事件驱动系统的逻辑行为，驱动系统可以在不同的模式之间进行切换。

6）Real-Time Workshop：直接从 Simulink 框图自动生成 C 或 Ada 代码，用于实现快速原型和硬件的仿真，整个代码的生成可以根据需要完全定制。

7）Simulink Blockset：专门为特定领域设计的 Simulink 功能模块集合，用户也可以利用已有的模块或自行编写的 C 和 MATLAB 程序建立自己的模块。

1.1.2 数字图像处理技术的内容与发展现状

数字图像处理是采用一定的算法对数字图像进行处理，以获得人眼视觉或者某种接收系统所需要的图像过程。数字图像处理的基础是数学，主要任务是进行各种算法设计和算法实现。

目前，数字图像处理技术已经在许多不同的应用领域得到重视，并取得了巨大成就。根据应用领域要求的不同，数字图像处理技术可以分为许多分支技术，重要的分支技术有：

（1）图像变换

图像阵列很大时，若直接在空域中处理，计算量将很大。为此，通常采用各种图像变换方法，如傅里叶变换、沃尔什变换、离散余弦变换、小波变换等间接处理技术，将空域处理转换到变换域处理，这样可以有效地减少计算量，提高处理性能。

（2）图像增强与复原

它的主要目的是增强图像中的有用信息，削弱干扰和噪声，使图像更加清晰，或者将其转换为更适合人或机器分析的形式。图像增强并不要求真实地反映原始图像，而图像复原则要求尽量消除或减少获取图像过程中所产生的某些退化，使图像能够反映原始图像的真实面貌。

（3）图像压缩编码

在满足一定保真度的条件下，对图像信息进行编码，可以压缩图像的信息量，简化图像的表示，从而大大压缩图像描述的数据量，以便存储和传输；图像压缩在不同的应用背景下可以采取不失真压缩和失真压缩。

（4）图像分割

图像分割是数字图像处理中的关键技术之一，是为了将图像中有意义的特征提取出来。它是进一步进行图像识别、分析和理解的基础。图像中有意义的特征包括图像对象的边缘、区域等。

（5）图像识别

图像识别属于模式识别的范畴，其主要内容是在图像经过某些预处理（增强、复原、压缩）后，进行图像分割和特征提取，从而进行判别识别。图像识别常用的经典识别方法有统

计模式识别和句法模式识别。近年来，新发展起来的模糊模式识别和人工神经网络模式识别在图像识别中也越来越受到重视。

以上数字图像处理内容也并非孤立存在的，往往相互联系，而一个实用的图像处理系统也通常需要将几种图像处理技术结合起来，才能得到所需要的结果。例如，图像变换是图像编码技术的基础，而图像增强与复原一般又是图像处理的最终目的，也可作为进一步进行图像处理工作的准备；通过图像分割得到的图像特征既可以作为最后结果，也可以作为下一步图像分析的基础。



不同的图像处理技术应用于不同的领域，发展出了不同的分支学科，如遥感图像处理、医学图像处理等，其他如计算机图形学、模式识别、人工智能和机器人视觉等学科领域也与图像处理有密切的关系。

图像处理技术的发展大致经历了初创期、发展期、普及期和实用化期 4 个阶段。初创期开始于 20 世纪 60 年代，当时的图像采用像素型光栅进行扫描显示，大多采用中、大型机对其处理。在这一时期，由于图像存储成本高、处理设备昂贵，其应用面很窄。进入 20 世纪 70 年代的发展期，开始大量采用中、小型机进行处理，图像处理也逐渐改用光栅扫描显示方式，特别是 CT 和卫星遥感图像的出现，对图像处理技术的发展起到了很好的推动作用。到了 20 世纪 80 年代，图像处理技术进入普及期，此时的微机已经能够担当起图形图像处理的任务。超大规模集成电路（Very Large Scale Integration，VLSI）的出现更使处理速度大大提高，设备造价也进一步降低，极大地促进了图形图像系统的普及和应用。20 世纪 90 年代是图像处理技术的实用化时期，图像处理的信息量巨大，对处理速度的要求极高。

针对现有的实际应用，数字图像处理具有以下特点。

1）信息量大，要求处理速度比较快。目前，数字图像处理的信息大多是二维信息，处理信息量很大。例如，一幅 256 像素×256 像素低分辨率的黑白图像，要求约 64kbit 的数据量；对 512 像素×512 像素高分辨率的彩色图像，则要求 256kbit 的数据量；如果要处理 30f/s 的视频图像，则每秒要求处理 500kbit～22.5Mbit 的数据量。因此，对计算机的计算速度、存储容量等要求较高。

2）占用频带较宽。与语音信息相比，数字图像占用的频带要大几个数量级。如电视图像的带宽约 56MHz，而语音带宽仅为 4kHz 左右。所以，数字图像在成像、传输、存储、处理和显示等各个环节的实现上，技术难度较大，成本高，且对频带压缩技术提出了更高的要求。

3）数字图像中各个像素间的相关性强，压缩潜力大。在图像画面上，经常有很多像素有相同或接近的灰度。就电视画面而言，同一行中相邻两个像素或相邻两行间的像素，其相关系数可达 0.9 以上。一般而言，相邻两帧之间的相关性比帧内相关性还要大。因此，图像处理中的信息压缩潜力巨大。

4）图像质量评价受主观因素影响。数字处理后的图像一般需要给人观察和评价，而人的视觉系统很复杂，受环境条件、视觉性能、人的情绪、爱好以及知识状况影响很大，因此评价结果受人的主观因素影响较大。为此，如何客观评价图像质量还有待进一步深入研究。另外，计算机视觉是模仿人的视觉，人类的感知原理必然严重影响计算机视觉的研究。

5）图像处理技术综合性强。数字图像处理技术中涉及的基础知识和专业技术相当广泛，通常涉及通信技术、计算机技术、电子技术、电视技术以及更多的数学、物理等方面的

基础知识。例如，图像编码的理论基础是信息论和抽象数学的结合，而图像识别则需要掌握随机过程和信号处理方面的知识。此外，不少课题还需要更加专业的知识，如小波变换、神经网络、分形理论等。

另外，数字图像处理是一门应用性很强的学问，必须与计算机技术的发展相适应。随着电子技术和计算机技术的不断提高和普及，数字图像处理进入高速发展时期，目前 2GHz 以上的 CPU 已经开始推广应用，这将大大促进数字图像处理技术的发展。

1.2 相关学科和领域

1.2.1 数字信号处理学

数字信号处理学（Digital Signal Processing）是指用数字电路和数字计算机对信号进行数字化、滤波等处理，最典型的信号如电压、电流等随着时间变化的一维物理量。数字信号处理学的研究内容包括数字化原理和采样定理、数字滤波器、数字正交变换、数字信号编码压缩与传输等内容，其中最重要的概念包括傅里叶变换、频率、频谱、滤波器等。

数字图像是二维的数字信号，是随空间坐标变化的灰度值或颜色值，图像处理是指用数字电路和数字计算机对图像进行处理。因此，数字图像处理学也包括数字化原理和采样定理、图像滤波器、图像正交变换、图像编码和压缩与传输等内容。

由此可见，数字信号处理与数字图像处理是紧密相关的学科，数字图像处理是数字信号处理理论的二维扩展，数字信号处理理论中的进展会导致数字图像处理的新理论和方法，而数字图像处理的进展和应用又反过来对数字信号处理提出了更高的理论研究要求。

1.2.2 计算机图形学

计算机图形学（Computer Graphics）是指用计算机来实现图形的生成、表示、处理和显示。计算机图形学的研究内容包括物体或模型的数学模型、图形生成、几何透视变换、消隐（消去隐藏面）、覆盖表面纹理、光照模型和光线跟踪等。

计算机图形学通常是由数学公式经过计算，最终生成物体或模型的二维或三维仿真图形（逼真的图形可与实际图像媲美）；而数字图像处理则通常是由数字图像数据进行处理，最终识别出图像中的景物，甚至得到景物的统计参数和数学模型。因此，图形学和图像学是互逆的处理过程，二者是有本质区别的。

早期的图形通常是指由点、线、面等元素来表达的三维物体，现代计算机图形学则可以生成完全逼真的图像，再加上计算机图形学的设备也采用几乎相同的图像输入、输出和显示设备，导致人们把图形和图像的称谓混淆。也就是说，图的共性和图形的共性，容易引起图形和图像这两个概念的混淆，这是需要注意的，也是可以理解的。

1.2.3 计算机视觉

计算机视觉（Computer Vision）是研究计算机感受和理解自然景物的理论和技术，也可以是研究机器人感受和理解自然景物的理论和技术，所以也称为机器视觉（Machine Vision）。计算机视觉的研究和设计目的是依照人类或动物的视觉系统，开发出能够感觉和理

解自然景物的计算机和机器人视觉系统。

视觉是人类观察世界、认知世界的重要功能和手段。人类从外界获得的信息约有75%来自视觉系统，这既说明视觉信息量巨大，也表明人类对视觉信息有较高的利用率。人类视觉过程是一个从感觉（感受图像：三维世界的二维平面投影）到认知（分析图像：由二维图像推断和理解三维世界的内容及相互关系）的复杂过程。视觉的目的是要对场景做出对观察者有意义的理解和描述，并可以根据周围环境的不同和观察者的意愿进行相应的反应和动作。

计算机视觉是指用计算机实现人的视觉功能，对客观世界的三维场景进行感知、识别和理解。因此，计算机视觉也研究数字图像和数字图像处理，但其研究重点在于视觉的立体成像的原理、图像处理方法及实现，或视觉动图像的成像原理、处理方法及实现。因此，计算机视觉与数字图像处理是紧密相关的学科领域，二者相互促进、相互依赖和相互补充。

1.3 MATLAB 2007a 的新功能

1.3.1 MATLAB 2007a 的新特性

MATLAB 是一种高级科学计算语言，是进行数据分析与算法开发的集成开发环境。MATLAB 2007a 针对编程环境、代码效率、数据可视化、数学计算、文件 I/O 操作等方面都进行了升级，增加了新功能。为此，相对于以前的版本，MATLAB 2007a 具有一些新的特性。

1. 开发环境方面

1）重新设计的桌面环境，针对多文档界面应用提供了简便的管理和访问方法，允许用户自定义桌面外观，创建常用命令的快捷方式。

2）增强数组编辑器（Array Editor）和工作空间浏览器（Workspace Browser）功能，用于数据的显示、编辑和处理。

3）在当前目录浏览器（Current Directory Browser）工具中，增加了代码效率分析、覆盖度分析等功能。

4）增加了 M-Lint 编码分析，能辅助用户完成程序性能分析，提高程序的执行效率。

5）增强了 M 文件编辑器（M-Editor）的功能，可以支持多种格式的源代码文件可视化编辑，如 C/C++、HTML、Java 等。

2. 编程方面

1）新增支持创建嵌套函数（Nested Function）功能，并且提供了更加灵活的代码模块化方式。

2）具有匿名函数（Anonymous Function）功能，支持在命令行或脚本文件中创建单命令行函数（Single Command Line Function）。

3）具有模块化注释功能，可为整个代码段进行注释。

3. 数学运算方面

1）支持整数算术运算和单精度数据类型运算，包括基本算术运算、线性代数、FFT 操

作等。

2）使用了更强大的计算算法包 Qhull 2002. 1，支持更丰富的计算算法。

3）利用专门的 linsove()函数处理线性代数方程的求解问题。

4）提供了 ODE 求解器，用于处理隐性微分方程组及多点边界问题。

4．图形和 3D 可视化方面

1）对图形窗体界面进行了更新，并且能直接从图形窗体生成 M 代码，并完成用户自定义绘图。

2）增强了图形窗体的注释功能。

3）提供了数据侦测工具（Data Detection Tools），更加方便了数据观测与跟踪。

4）支持自定义图形对象，并提供了丰富的图形显示能力。

5）GUIDE 新增了对用户界面面板和 ActiveX 控件的支持。

6）增强了句柄图形对象，可以支持完整的 TeX 和 LaTeX 字符集。

5．文件 I/O 和外部接口方面

1）新增文件 I/O 函数，支持读取任意格式文本数据文件，并且支持写入 Excel 和 HDF5 格式数据文件。

2）MAT 文件格式具有压缩功能，可进行快速数据文件的 I/O 操作。

3）使用 javaaddpath()函数，无须重新启动 MATLAB 就可以实现 Java 类的加载、删除等功能。

4）支持 COM、服务器事件以及 VBS。

5）支持 SOAP，使用网络服务。

6）支持 FTP 对象，可以直接访问 FTP 服务器。

7）支持 Unicode 编码格式，增强了 MAT 文件字符集。

6．性能与系统平台支持

1）使用了 JIT 加速器，支持所有数值数据类型。

2）在 Windows XP 操作系统下，可以支持 3GB 内存访问。

1.3.2 Simulink 6 的新特性

Simulink 是交互式动态系统建模、仿真和分析的图形环境，是进行基于模型的嵌入式系统开发的基础开发环境，可以用于控制系统、信号处理以及通信系统等的建模、仿真和分析等工作。Simulink 6 相对于以前的版本，改善了性能，并且针对大规模的系统开发进行了性能优化，主要体现在以下方面。

1．大系统建模

1）支持将大系统模型分割为不同的文件，且每个文件为独立的系统模型。

2）支持对系统的不同模型文件进行独立的仿真测试。

3）增强了系统的集成功能，支持配置管理和版本控制。

4）新增了递增式模型加载与代码生成功能。

5）针对大系统模型，提高了运行速度和执行效率。

6）对于模型工作空间（Model Workspace），每个模型都可以有独立的工作空间，可以实现独立的参数管理和数据处理。

7）增强了总线的支持能力。

2. Simulink 与 Sateflow

1）统一的模型浏览器（Model Browser），用于浏览、维护、配置、搜索、定义所有模型中相关的参数、数据对象和属性。

2）统一的仿真和代码生成选项。

3）支持创建并保存多种仿真和代码生成选项。

4）数据管理和可视化。

5）新增数据对象属性。

6）可选数据记录增加测试点，无需在模型中增加额外的模块。

7）I/O 管理，可以将必要的信号源和信宿连接到模型而不需要增加模块。

1.4 MATLAB 2007a 图像处理

由于图像操作很多，这里仅仅以 MATLAB 窗口的操作、图像的噪声消除和边缘检测为例，来说明该工具的基本使用方法。

1.4.1 MATLAB 图像处理应用举例

1. MATLAB 的打开

运行安装完成的 MATLAB 程序，若在桌面上建立了 MATLAB 的快捷方式，单击桌面的 MATLAB 图标即可启动 MATLAB，其 Command Window 窗口如图 1-1 所示。

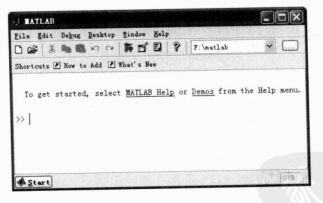


图 1-1 MATLAB 的 Command Window 窗口

2. 图像输入到计算机

将所需要处理的图像通过数码相机、U 盘等输入设备输入到计算机中，并确定图像在计算机中存放的位置。例如，图像存放在 D:\MATLAB\work 中。

3. 打开编辑窗口编写程序

启动 MATLAB 的 M 文件编辑器，可以在 Command Window 窗口中选择 File→New→M→File 命令，即可建立. M 文件。在编辑窗口中输入要处理图像的源程序，如图 1-2 所示。

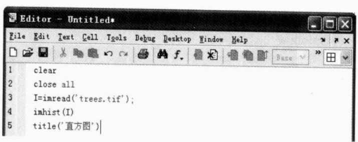


图 1-2　编辑程序

4. 保存并运行

选择 Debug→Save and Run 命令，即可运行程序并保存该程序。程序运行结果如图 1-3 所示。

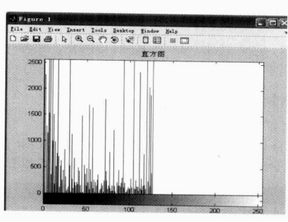


图 1-3　程序运行结果

5. 保存运行结果

若想将运行结果保存成图片的格式，须在程序中加入图像 I/O 文件中的图像写入图形文件函数，即 imwrite（A，filename，fmt）。若想让图像 I 保存在 D 盘 TAB 文件夹中，文件名为 123，文件格式为 BMP，可用语句“imwrite (I，'D:\TAB\123. BMP')”。

1.4.2 图像处理基本操作

1. 读取图像并显示

在读取图像之前，应该首先清除 MATLAB 所有的工作平台变量，并关闭打开的图形窗口。为此，可使用以下命令行：

```
clear;   close all
```

然后使用图像选取 imread()函数就可以读取一幅图像。假设 1.4.1 节读取图像为 trees. tif（该图像是图像处理工具箱自带的图像），并将它存储在一个名为 I 的数组中，以上可以使用命令：

```
I=imread('trees. tif');
```

然后可以调用 imshow 命令来显示图像：

```
imshow(I)
```

得到的显示结果如图 1-4 所示。



图 1-4 读取的原始图像

2. 检测内存中的图像

使用如下命令可以查看图像 I 的存储方式：

```
whos
```

运行后输出结果如下：

```
Name        Size                 Bytes   Class
I        258×350                 90300   uint8 array
Grand total is 90300 elements using 90300 bytes
```

以上输出结果说明图像采用 8 位存储方式，并占用了 90300 B 的存储空间。

3. 实现直方图均衡化

由图 1-4 可知，trees. tif 对比度比较低，为了更好地观察图像的灰度分布信息，可以用 imhist()函数创建描述图像灰度分布的直方图，并使用 figure 命令将直方图显示在一个新的图像窗口，如图 1-3 所示。

```
figure, imhist (I)   %在新图中显示图像 I 的直方图
```

从图 1-3 中可以看出，由于图像的灰度范围比较狭窄，没有覆盖整个灰度范围[0, 255]（图像的默认存储类型为 unit8），并且图像中灰度值的高低区分不明显，因而不能产生好的对比效果。要产生较好的对比效果，可以调用 histep()函数将图像的灰度值扩展到整个灰度范围中，从而将数组 I 的对比度提高。修改过的图像数据将保存在变量 I2 中，并在一个新的图像窗口中显示均衡处理的图像 I2。利用以下命令行，可得到如图 1-5 所示的处理结果。

```
I=imread( 'trees. tif' );
I2=histeq( I );
figure,imshow( I2 )
```

对图像 I2，再调用 imhist()函数可以观察均衡后图像的灰度值分布情况，如图 1-6 所示。

```
figure,imhist(I2)
```

通过使用 histeq()函数来调节图像的像素分布，使之能够分布在与图像类型有关的整个取值范围内。对于一幅图像，如果存储类型是 unit8，那么相应的取值范围就是[0, 256]；如果存储类型是 unit16，则取值范围是[0, 655351]；如果是双精度类型，则取值范围是[0, 1]。

比较图 1-3 和图 1-6 可知，I2 的直方图比 I 的直方图要长且平坦，这种铺展直方图的过程就叫直方图均衡化。


图 1-5　均衡后图像

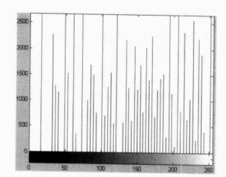


图 1-6　均衡后图像的灰度直方图

4．保存图像

下面，将把均衡化后的图像 I2 保存到磁盘中，如果希望将该图像保存为 PNG 图像文件格式，则可以使用 imwrite()函数，并指定该保存图像的文件名和文件的扩展名.png，程序代码如下：

```
imwrite (I2,'trees2. png');
```

5．检查新生成文件的内容

保存图像后，如何知道磁盘上写了什么内容呢？可以使用 imfinfo()函数来观察保存的图像文件信息。需要注意的是，在使用 imfinfo()函数时，不能在命令行末尾加上分号，这样才可以确保 MATLAB 能够显示图像输出结果（只有 imfinfo()函数如此，其他显示函数加不加分号并没有影响，以下不再说明）。此外，还须保证此时的文件路径与调用 imwrite()函数时的路径一致，如以下命令行所示。

```
imfinfo('trees2. png')
```

系统得到响应：

```
ans =
    Filename: 'trees2. png'
    FileModDate:'10-Mar-2008 00:00:26'
    FileSize:55937
    Format:'png'
    FormatVersion: []
    Width: 350
    Height:258
    BitDepth: 8
    ColorType: 'grayscale'
    FormatSignature: [137 80 78 71 13 10 26 10]
```

1.4.3 图像处理的高级应用

在上面的操作中，读者对 MATLAB 中的一些基本操作已有了一定的了解，通过以下的练习，将掌握 MATLAB 的一些较为高级的操作。本练习的主要目的是消除 rice. tif 图像中亮

度不一致的背景，并使用阈值将修改后的图像转换为二值图像，使用成员标记返回图像中对象的个数以及统计特性。

1. 读取和显示图像

清除 MATLAB 工作平台的所有变量，关闭已经打开的图形窗口，读取和显示灰度图像 rice. png。

```
clear,close all
I=imread('rice. png');
imshow(I)
```

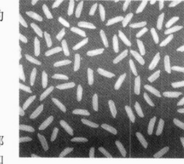
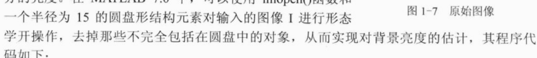


图 1-7 原始图像

2. 估计图像背景

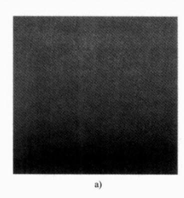
如图 1-7 所示，图像中心位置的背景亮度要高于其他部分的亮度。在 MATLAB 7.0 中，可以使用 imopen()函数和一个半径为 15 的圆盘形结构元素对输入的图像 I 进行形态



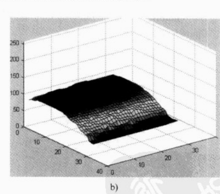
学开操作，去掉那些不完全包括在圆盘中的对象，从而实现对背景亮度的估计，其程序代码如下：

```
background=imopen(I,strel('disk',15));
imshow(background)
figure,surf(double(background(1:8:end,1:8:end))),zlim([0 256]);
set(gca,'Ydir','reverse');
```

生成的背景图如图 1-8a 所示。图 1-8b 为以表面图形式显示的背景。



a)



b)

图 1-8 背景及背景表面图

a) 背景图 b) 背景表面图

3. 从原始图像中减去背景图像

现在将背景图像 background 从原始图像 I 中减去，从而创建一个新的、背景较为一致的图像，如图 1-9 所示。

```
I2=imsubtract (I,background );
figure,imshow( I2 )
```



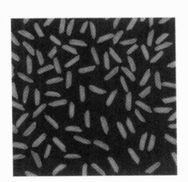


图 1-9　除去背景后的图像

4．调节图像对比度

从图 1-9 可以看出，修改后的图像比较暗，可以使用 MATLAB 提供的 imadjust()函数来调节图像的对比度，程序代码如下：

```
I3=imadjust(I2,stretchlim(I2),[0 1]);
figure,imshow(I3)
```

调节后的图像效果如图 1-10 所示。

5．使用阈值操作将图像转换为二进制图像

使用 graythresh()和 lm2bw()函数可以创建一个新的二值图像 BW，如图 1-11 所示。

```
level=graythresh(I3);
BW=im2bw(I3,level);
figure,imshow(BW)
```



图 1-10　调节对比度后的图像



图 1-11　二值化图像

调用 whos 命令可以查看图像的存储信息：

```
>> whos
  Name        Size           Bytes    Class
  BW          256×256        65536    logical array
  I           256×256        65536    uint8 array
  I2          256×256        65536    uint8 array
  I3          256×256        65536    uint8 array
```

```
background      256×256          65536   uint8 array
level           1×1              8   double array
Grand total is 327681 elements using 327688 bytes
```

6. 检查图像中的对象个数

为了确定图像中的米粒个数，可以使用 bwlabel()函数。该函数表示了二值图像 BW 中的所有相关成分，并且返回在图像中找到的对象个数。

```
[labeled,numObjects]=bwlabel(BW,4)
numObjects =
        101
```

对于图 1-11 中有些相互连接的米粒，bwlabel()函数则将它们视为同一个对象。

7. 检查标记矩阵

可以使用 imcrop()函数来选择并显示已标记的对象和部分背景内的像素。选择一个较小的矩形来进行这操项作，以保证显示的像素值不会引起 MATLAB 命令窗口的滚动。以下语句将使用 imcrop()函数进行交互式操作，当用户的鼠标位于图像范围内，其形状会变成十字形，通过单击鼠标并进行拖动来选择一个标记区域，选择完成后，imcrop()函数将显示用户指定的标记区域（根据选择区域的不同，grain 显示的矩阵是不同的）。

```
grain = imcrop(labeled)
grain =
     0     0     0    14
     0     0     0    14
     0     0     0     0
     0     0     0     0
```

观察标记矩阵的一个办法就是将其显示为一幅伪彩色的索引色图像。在伪彩色的彩色图像中，标记矩阵中的每一个对象都将被映射为相关调色板中的不同颜色，可以使用 label2rgb 色板中的对应颜色，处理效果如图 1-12 所示。

```
RGB_label=label2rgb(labell ed,@spring,'c','shuffle');
imshow(RGB_label)
```

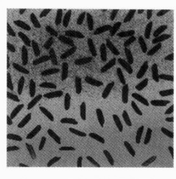


图 1-12　伪彩色索引色图像



第 2 章　图像的编码和解码

2.1　概述

2.1.1　图像压缩编码的必要性

数字图像已经在人们的生活中随处可见，并且成为生活中不可缺少的一部分。但是图像数字化之后，其数据量是非常庞大的。例如，一幅 640 像素×480 像素分辨率的彩色图像（24bit/像素），其数据量为 900KB。如果以 30f/s 的速度播放，则每秒的数据量为 640×480×24×30bit =210.9Mbit=26.4MB，需要 210.9Mbit/s 的通信回路；如果存放在 650MB 的光盘中，在不考虑音频信号的情况下，每张光盘也只能播放 24s。毫无疑问，如果不进行编码压缩处理，在图像存储中所遇到的困难和成本之高是可想而知的。对于利用电话线传送黑白二值图像的传真，如果以 200dpi（点/英寸）的分辨率传输，一张 A4 稿纸内容的数据量为（200×210/25.4）×（200×297/25.4）bit=3866948bit，按目前 14.4kbit/s 的电话线传输速率，需要传送的时间是 263s（4.4min）。图 2-1 为压缩后的图像。



图 2-1　原图像数据为 8.5MB，压缩后（Buaa.jpg）为 0.98MB

总之，大数据量的图像信息会给存储器的存储容量、通信干线信道的带宽以及计算的处理速度增加极大的压力。单纯靠增加存储容量，提高信道带宽以及计算机的处理速度等方法来解决这个问题是不现实的，这时就要考虑压缩。因此，图像数据在传输和存储中，数据的压缩都是必不可少的。

2.1.2　图像压缩编码的可能性

图像压缩的理论基础是信息论。从信息论的角度来看，压缩就是去掉信息中的冗余，即保留不确定的信息，去掉确定的信息（可推知的），也就是用一种更接近信息本质的描述来代替原有冗余的描述。一幅图像存在着大量的数据冗余和主观视觉冗余，因此图像数据压缩

既是必要的，也是可能的。

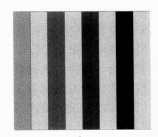
1．数字图像本身的特征带来数据压缩的可能性

（1）空域冗余

空域冗余也称为空间冗余或几何冗余，空域冗余是一种与像素间相关性直接联系的数据冗余。通常邻近像素灰度分布的相关性很强。例如，图 2-2a 所示的条状物体排列比较整齐，如果用水平方向的任何一行像素预测垂直方向的其他行像素，都能够准确预测其他行数据，其他行数据完全能够用一行数据复制得到。图 2-2b 中也有 4 个相同的条状物体，但是随便摆放，就不能用图 2-2b 中某一行预测其他行。我们称图 2-2a 中数据存在较大冗余，能够给图像压缩提供较大压缩空间。

（2）时域冗余

时域冗余又称时间冗余，它是针对视频图像而言的。视频序列为 25～30f/s 的图像，连续播放，相邻帧之间的时间间隔很小（例如，25f/s 的电视信号，其帧间时间间隔只有 0.04s）；同时，实际生活中的运行物体具有运行一致性，使得视频序列图像之间有很强的相关性。



a)　　b)

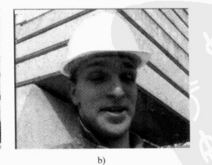
图 2-2　图像的空域冗余

a) 比较整齐条形物　b) 随便摆放条形物

例如，图 2-3a 是一组视频序列的第 1 帧，图 2-3b 是第 2 帧。人眼很难发现两帧图像的差别，如果视频连续播放，人眼就更难看出两帧图像有差别了。两帧图像越接近，说明图像携带的信息越少。换句话说，第 2 帧相对第 1 帧而言，存在大量冗余。对于视频压缩而言，时域冗余是视频图像压缩中可利用的最主要的冗余。



a)



b)

图 2-3　图像的时域冗余

a) 第 1 帧视频图　b) 第二帧视频图

（3）频域冗余

频域冗余是针对目前普通使用的变换编码而言的。绝大部分变换编码将空域的图像变换到频域中，使得大量的信息能用较少的数据来表示，从而达到压缩的目的。从空域转化到频域，去除了图像像素在空域的相关性，然而人们发现，图像在频域的表示（频谱系数），同样存在冗余。大多数图像的频谱具有低通特性，低频部分的系数能够提供绝大部分的图像信息，保留低频部分的系数，而丢弃高频部分的系数，因此保持了大部分图像能量，在恢复图像时带来的质量劣化并不明显，去除频域的冗余，能够进一步提升压缩的空间，可提高编码效率。

（4）信息熵冗余

图像中像素灰度出现的不均匀性，会造成图像信息熵冗余，即用同样长度比特表示每一个灰度，则必然存在冗余。若将出现概率大的灰度级用长度较短的码表示，将出现概率小的灰度级用长度较长的码表示，有可能使编码总长度下降。

2. 应用环境允许图像有一定程度的失真

1）接收端图像设备分辨率较低，则可降低图像分辨率。

2）用户所关心的图像区域有限，可对其余部分图像采用空间和灰度级上的粗化。

3）根据人的视觉特性对不敏感区域进行降分辨率编码（视觉冗余）。

可以利用视觉的这一特点编码去除人眼的视觉冗余。通常，人眼能够分辨的灰度级有限，同时，它所感受的图像区域物体的亮度不仅仅与物体的反射光有关，还具有马赫带效应、同时对比度、视觉暂留及视觉非线性等特点，有些信息在通常的视觉感知过程中并不那么重要，这些信息可被认为是视觉冗余，去除这些冗余，人眼不会明显地感受到图像质量的降低，视觉冗余也给图像压缩提供了可能。

2.1.3 图像压缩编码的评价准则

在图像压缩编码中，解码图像与原始图像可能会有差异，因此，需要评价压缩后图像的质量。描述解码图像相对原始图像偏离程度的测度一般称为保真度（逼真度）准则。常用的准则可分为客观保真度准则和主观保真度准则两大类。

1. 客观保真度准则

最常用的客观保真度准则是原始图像和解码图像之间的方均根误差和方均根信噪比两种。令 $f(x,y)$ 代表原图像，$\hat{f}(x,y)$ 代表对 $f(x,y)$ 先压缩又解压缩后得到的 $f(x,y)$ 的近似值，对任意 x 和 y，$f(x,y)$ 和 $\hat{f}(x,y)$ 之间的误差定义为

$$e(x,y)=\hat{f}(x,y)-f(x,y) \tag{2-1}$$

若 $f(x,y)$ 和 $\hat{f}(x,y)$ 均为 $M\times N$，则它们之间的方均根误差 e_{rms} 为

$$e_{\text{rms}}=\left\{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y)-f(x,y)]^2\right\}^{\frac{1}{2}} \tag{2-2}$$

如果将 $\hat{f}(x,y)$ 看做原始图像 $f(x,y)$ 和噪声信号 $e(x,y)$ 的和，那么解压图像的方均信噪比 SNR_{ms} 为

$$SNR_{ms}=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}^2(x,y)\Big/\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y)-f(x,y)]^2 \tag{2-3}$$

如果对上式求平方根，就得到方均根信噪比 SNR_{rms}。实际使用中常将 SNR_{ms} 归一化并用分贝（dB）表示，令

$$\bar{f}=\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y) \tag{2-4}$$

则有

$$SNR=10\lg\left\{\frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[f(x,y)-\bar{f}]^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y)-f(x,y)]^2}\right\} \tag{2-5}$$

如果令 $f_{max}=\max f(x,y), x=0,1,\cdots,M-1, y=0,1,\cdots,N-1$，则可得到峰值信噪比 $PSNR$：

$$PSNR=10\lg\left\{\frac{f_{max}^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y)-f(x,y)]^2}\right\} \tag{2-6}$$

2. 主观保真度准则

尽管客观保真度准则提供了一种简单、方便的评估信息损失的方法，但很多解压图像最终是供人观看的。事实上，具有相同客观保真度的不同图像，人的视觉可能产生不同的视觉效果。这是因为客观保真度准则是一种统计平均意义下的度量准则，对于图像中的细节无法反映出来。而人的视觉系统具有独特的特性，能够觉察出来。这种情况下，用主观的方法来测量图像的质量更为合适。一种常用的方法是对一组（不少于 20 人）观察者显示图像，并将他们对该图像的评分取平均，用来评价一幅图像的主观质量。

评价也可以对照某种绝对尺度进行。表 2-1 给出一种对电视图像质量进行绝对评价的尺度，根据图像的绝对质量进行判断打分。也可通过将 $f(x,y)$ 和 $\hat{f}(x,y)$ 比较并按照某种相对的尺度进行评价。如果观察者将 $f(x,y)$ 和 $\hat{f}(x,y)$ 逐个进行对照，则可以得到相对的质量分。例如，可用{-3，-2，-1，0，1，2，3}来代表主观评价{很差，较差，稍差，相同，稍好，较好，很好}。

表 2-1 电视图像质量评价尺度

评 分	评 价	说 明
1	优秀	图像质量非常好，如同人能想像出的最好质量
2	良好	图像质量高，观看舒服，有干扰但不影响观看
3	可用	图像质量可以接受，有干扰但不太影响观看
4	刚可看	图像质量差，干扰有些妨碍观看，观察者希望改进
5	差	图像质量很差，妨碍观看的干扰始终存在，几乎无法观看
6	不能用	图像质量极差，不能使用

2.2 统计编码

2.2.1 信息熵

信息是消息的不确定性度量，消息的可能性愈小，其蕴含的信息就愈多，即不确定程度愈大；反之，消息的可能性愈大，则其信息愈少，即不确定程度愈小。

假设从 N 个数中选中某个数 x 的概率为 $P(x)$，则根据 Shannon（香农）理论，定义其信息为

$$I(x) = -\log_2 P(x) \tag{2-7}$$

如果将信源所有可能事件的信息量进行平均，就得到信息熵（entropy），所谓熵就是平均信息量。

信源 x 的符号集为 $x_i (i = 1,2,\cdots,n)$，设 x_i 出现的概率为 $P(x_i)$，则信源 x 的熵为

$$H(x) = \sum_{i=1}^{n} P(x_i) I[P(x_i)] = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{2-8}$$

根据 Shannon 无噪声编码定理，对于熵为 H 的信源，对其无失真编码，所能达到的最大值为 $(H+\varepsilon)$ bit/每符号，这里 ε 为一任意小的正数，因此可能达到的最大压缩比为

$$C_{\max} = \frac{B}{H+\varepsilon} \approx \frac{B}{H} \tag{2-9}$$

式中，B——原始图像的平均比特率，且定义压缩比为

$$C = \frac{\text{原始数据的平均比特率}(B)}{\text{压缩数据的平均比特率}(H)} \tag{2-10}$$

方均信噪比为

$$SNR_{\text{ms}} = \frac{\sigma_u^2}{\sigma_e^2} \tag{2-11}$$

方均根信噪比为

$$SNR_{\text{rms}} = \sqrt{SNR_{\text{ms}}} \tag{2-12}$$

2.2.2 Shannon Fano 编码

一种常用的变长编码是 Shannon Fano 编码，这种编码有时也可以得到最优编码性能。它的编码准则要符合非等长条件，在码字中 1 和 0 是独立的，而且是（或差不多是）等概率的。这样的准则一方面可保证无需用间隔来区分码字，同时又保证每传输 1 位码就蕴含着 1 队的信息量。

为定义有效的代码表，设计了一个特定算法——Shannon Fano。编码就是根据该算法建立的。其主要准则是：在码字中，1 与 0 是独立的，而且几乎是等概率。实际算法如下：

1）对于一个给定的符号序列，对应一个概率或频率记数序列，使每个符号的相对出现频率已知。

2）根据频率对符号进行排序，频率最高的在顶部，频率最低的在底部。

3）将序列分成两部分，上半部分的频率之和尽可能接近下半部分的频率之和。

4）序列的上半部分赋给二进制数字 0，而下半部分赋给二进制数字 1。

5）对所分的两部分，分别重复使用第 3）步和第 4）步，直到不能再分为止。具体实现过程如图 2-4 所示。

由图 2-4 可知，出现频率高的字符在编码中只占有较少的位数，这显然是合理的，因为一个给定字符的信息容量公式是字符出现概率以 2 为底的对数的负值。





图 2-4　Shannon Fano 编码实现

2.2.3 哈夫曼编码

哈夫曼（Huffman）编码是哈夫曼树的一个应用。哈夫曼编码应用广泛，如 JPEG 中就应用了哈夫曼编码。

哈夫曼树又称为最优二叉树，是一种加权路径长度最短的二叉树。所谓树的加权路径长度，就是树中所有叶节点的权值乘上其到根节点的路径长度，若根节点为 0 层，叶节点到根节点的路径长度为叶节点的层数，记为 $WPL=(W_1\times L_1+W_2\times L_2+W_3\times L_3+\cdots+W_n\times L_n)$ 表示 N 个权值 $W_i(i=1,2,\cdots,n)$ 构成一棵有 n 个叶节点的二叉树，相应的叶节点的路径长度为 $L_i(i=1,2,\cdots,n)$。可以证明哈夫曼树的 WPL 是最小的。

哈夫曼编码是 20 世纪 50 年代初由哈夫曼提出的，它根据字符出现的概率来构造平均长度最短的编码，是一种变长编码，如果各码字长度严格按照码字所对应的符号出现概率大小的逆序排列，则编码的平均长度最小。需要说明的是，码字即为符号经哈夫曼编码后得到的编码，其长度因符号出现的概率的不同而不同，因此是变长度的。图 2-5 是哈夫曼编码实例。

哈夫曼编码的编码步骤如下：

1）将信源符号出现的概率按递减的顺序排列。

2）将两个最小的概率进行组合相加，并重复该步骤，始终将较高的概率分支放在上

部，直到概率等于 1 为止。

3）对每一个组合中的元素进行 1 位码字分配，如概率较大的元素分配码字 1，概率较小的元素分配码字 0，反向分配也可以。

4）对每个信源符号从右到左排列，分配码字序列，得到非等长度的哈夫曼编码。

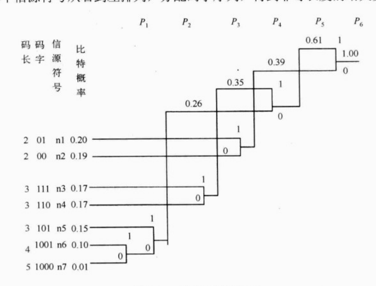


图 2-5 哈夫曼编码实例

实际应用中，由于在哈夫曼编码之前需要知道信源数据符号（叶节点）的概率，则为要求做实时编码的作业带来了麻烦。因此，目前在实时编码作业中，大多采用所谓准可变字长码，例如，采用双字长编码，并从短码集合中留出一个码子，作为长码字头，以保证码字的非续长特性。此外，在数字图像通信所用的三类传真机中的 MH 码，则采用了多字长 VLC 技术，它是根据一系列标准图像的统计分析结果，预先在其 IC 芯片中做好码表，使得实际的编码解码作业简化为一个查表过程，从而满足了高速实时处理的需要。

下面给出哈夫曼编码的 MATLAB 实现。

程序将输入的向量（矩阵）进行哈夫曼编码，然后反编码，判断是否是无失真编码，最后给出压缩前后的存储空间的比较。

```
clear all
fprintf('Reading data...')
data = imread('cameraman.tif');
data = uint8(data); %读入数据,并将数据限制为 uint8
fprintf('Done!\n')
%编码压缩
fprintf('compressing data ...');
[zipped,info] = norm2huff(data);
fprintf('Done!\n')
%解压缩
fprintf('compresing data ...');
```



```
unzipped = huff2norm(zipped,info);
fprintf('Done!\n')
%测试是否无失真
isOK = isequal(data(:),unzipped(:))
%显示压缩效果
whos data zipped unzipped
%%%%%%%%%%% norm2huff   %%%%%%%%%%
function [zipped,info] = norm2huff(vector)
if ~isa(vector,'uint8'),
      error('input argument must be a uint8 vector')
end
vector = vector(:)';
%将输入向量转换为行向量
f = frequency(vector);
%计算各元素出现的概率
simbols = find(f~=0);
f = f(simbols); %将元素按照出现的概率排列
[f,sortindex] = sort(f);
simbols = simbols(sortindex);
%产生码字 generate the codeword as the 52 bits of a double
len = length(simbols);
simbols_index = num2cell(1:len);
codeword_tmp = cell(len,1);
while length(f)>1,
      index1 = simbols_index{1};
      index2 = simbols_index{2};
      codeword_tmp(index1) = addnode(codeword_tmp(index1),uint8(0));
      codeword_tmp(index2) = addnode(codeword_tmp(index2),uint8(1));
      f = [sum(f(1:2)) f(3:end)];
      simbols_index = [{[index1 index2]} simbols_index(3:end)];
%将数据重新排列,使两个节点的频率尽量与前一个节点的频率相当
resort data in order to have the two nodes with lower frequency as
      first to
      [f,sortindex] = sort(f);
      simbols_index = simbols_index(sortindex);
end
%对应相应的元素与码字
codeword = cell(256:1);
codeword(simbols) = codeword_tmp;
%计算总的字符串长度
len = 0;
for index=1:length(vector),
      len = len+length(codeword{double(vector(index))+1});
end
%产生 01 序列
string = repmat(uint8(0),1,len);
```

```
pointer = 1;
for index=1:length(vector),
    code = codeword{double(vector(index))+1};
    len = length(code);
    string(pointer+(0:len-1)) = code;
    pointer =pointer+len;
end
%如果需要,加零
len = length(string);
pad = 8-mod(len,8);
if pad>0,
    string =[string unit8(zeros(1,pad))];
end
%保存实际有用的码字
codeword =codeword(simbols);
codelen =zeros(size(codeword));
weights = 2.^(0:23);
maxcodelen = 0;
for index 1:length(codeword),
    len = length(codeword{index});
    if len>maxcodelen,
        maxcodelen = len;
    end
    if len>0,
        code = sum(weights(codeword{index}= =1));
        code =bitset(code,len+1);
        codeword{index} = code;
        codelen(index) = len;
    end
end
codeword = [codeword{:}]
%计算压缩后的向量
cols = length(string)/8;
string = reshape(string,8,cols);
weights = 2.^(0:7);
zipped = unit8(weights*double(string));
%存储一个稀疏矩阵
huffcodes = sparse(1,1); % init sparse matrix
for index = 1:numel(codeword),
    huffcodes(codeword(index),1) = simbols(index);
end
%产生信息结构体
info.pad = pad;
info.ratio = cols./length(vector);
info.length = length(vector);
info.maxcodelen = maxcodelen;
```





```
%%%%%%%%%%      addnode   %%%%%%%%%%
function codeword_new = addnode(codeword_old,item)
codeword_new = cell(size(codeword_old));
for index =1:length(codeword_old),
    codeword_new{index} = [item codeword_old{index}];
end
%%%%%%%%%     huff2norm    %%%%%%%%%%%%%%
function vector =huff2norm (zipped,info)
% HUFF2NORM Huffman 解码器
% HUFF2NORM(X,INFO)根据信息结构体 info 返回向量 zipped 的解码结果%
%矩阵参数以 X(:)形式输入
if ~isa(zipped,'unit8'),
    error('input argument must be a unit8 vector')
end
%产生 01 序列
len = length(zipped);
string =repmat(uint8(0),1,len.*8);
bitindex = 1:8;
for index = 1:len,
    string(bitindex+8.*(index-1)) = uint8(bitget(zipped(index),bitindex));
end
%调整字符串
string = logical(string(:)'); % make a row of it
len =   length(string);
string ((len-info.pad+1):end) = []; % remove 0 padding
len = length(string);
%解码
weights = 2.^(0:51);
vector =repmat(uint8(0),1,info.length);
vectorindex = 1;
codeindex = 1;
code = 0;
for index = 1:len,
    code = bitset(code,codeindex,string(index));
    codeindex = codeindex+1;
    byte = decode(bitset(code,codeindex),info);
    if byte>0, %
        vector(vectorindex) = byte-1;
        codeindex = 1;
        code = 0;
        vectorindex = vectorindex+1;
    end
end
%%%%%%%%%%%%%%%%    decode   %%%%%%%%%%%
function byte = decode(code,info)
byte = info.huffcodes(code);
```

```
%%%%%%%%%%%%%%%  frequency  %%%%%%%%%%
function f = frequency(vector)
%FREQUENCY 计算元素出现概率
if ~isa(vector,'uint8'),
    error('input argument must be a uint8 vector')
end
f = repmat(0,1,256);
%扫描向量
len = length(vector);
for index =0:256, %
    f(index+1) = sum(vector==uint8(index));
end
%归一化
f = f./len;
```

运行上述程序，得到结果为

```
>>whos
  Name        Size              Bytes          Class
  data       256×256            65535          uint8 array
  unzipped     65535            65535          uint8 array
  zipped       57712            57712          uint8 array
Grand total is 65536 elements using 65536 bytes
```

其中压缩的信息结构体 info 为

```
Pad:7
Huffcodes:[108471   double]
Ratio:0.8806
Length:65535
Maxcodelen:16
```

2.2.4 算术编码

在 2.2.3 节中已经说明，哈夫曼编码使用整数个二进制位对符号进行编码，这种方法在许多情况下无法得到最优的压缩效果。假设某个字符的出现概率为 0.8，该字符事实上只需要$-\log_2(0.8)=0.322$ 位编码，但哈夫曼编码一定会为其分配一位 0 或一位 1 的编码。可以想象，整个信息的 80%在压缩后都几乎相当于理想长度的 3 倍左右，压缩效果可想而知。

难道真的能只输出 0.322 个 0 或 0.322 个 1 吗？算术编码就实现了这样的编码，而且自述编码在图像数据压缩标准（如 JPEG、H.264 等）中扮演了重要的角色。

在算术编码中，消息用 0～1 之间的实数进行编码，算术编码用到两个基本的参数：符号的概率和它的编码间隔。信源符号的概率决定压缩编码的效率，也决定编码过程中信源符号的间隔，而这些间隔包含在 0～1 之间。编码过程中的间隔决定了符号压缩后的输出。下面以一个简单的例子说明算术编码的编码过程。

假设信源符号为$\{00,01,10,11\}$，这些信源符号的概率分别为$\{0.1,0.4,0.2,0.3\}$。根据这些概率可把间隔$[0,1)$分成 4 个子间隔：$[0,0.1),[0.1,0.5),[0.5,0.7),[0.7,1)$，其中$[x,y)$表示半开放间隔，

即包含 x 不包含 y，见表 2-2。

表 2-2 信源符号、概率和初始编码间隔

符　号	00	01	10	11
概　率	0.1	0.4	0.2	0.3
初始编码间隔	[0,0.1]	[0.1,0.5]	[0.5,0.7]	[0.7,1]

如果二进制消息序列的输入为：10 00 11 00 10 11 01。编码时首先输入的符号是 10，找到它的编码范围是 $[0.5,0.7)$。由于消息中第二个符号 00 的编码范围是 $[0,0.1)$，因此它的间隔就取 $[0.5,0.7)$ 的第一个十分之一作为新间隔 $[0.5,0.52)$。依此类推，编码第三个符号是 11 时取新间隔为 $[0.514,0.52)$，编码第四个符号是 00 时，取新间隔为 $[0.514,0.5146)$，…。消息的编码输出可以是最后一个间隔中的任意数。整个编码过程如图 2-6 所示。

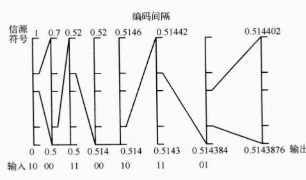


图 2-6 算术编码过程

根据以上例子，算术编码可以总结如下。

算术编码的基本思想是：算术编码不是将单个信源符号映射成一个码字，而是把整个信源表示为实数线上的 0～1 之间的一个区间，其长度等于该序列的概率，再在该区间内选择一个代表性的小数，转换为二进制作为实际的编码输出。消息序列中的每个元素都要用来缩短这个区间。消息序列中元素越多，所得到的区间就越小，当区间变小时，就需要更多的数位来表示这个区间。

由此可得，采用算术编码每个符号的平均编码长度可以为小数。

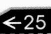
在算术编码中需要注意的几个问题：

1）由于实际中的计算机的精度不可能无限长，运算中出现溢出是一个明显的问题，但多数机器都有 16 位、32 位或者 64 位的精度，因此这个问题可使用比例缩放的方法解决。

2）算术编码器对整个消息只产生一个码字，这个码字是在间隔 $[0,1)$ 中的一个实数，因此译码器在接收到表示这个实数的所有位之前不能进行译码。

3）算术编码也是一种对错误很敏感的编码方法，如果有一位发生错误将导致整个消息译错。

算术编码可以是静态的或者自适应的。在静态算术编码中，信源符号的概率是固定的。在自适应算术编码中，信源符号的概率根据编码时符号出现的频繁程度动态地进行修改，在编码期间估算信源符号概率的过程叫做建模。由于很难事先知道精确的信源概率，



所以需要开发自适应的算术编码。当压缩消息时，不能期待一个算术编码器获得最大的效率，所能做的最有效的方法是在编码过程中估算概率。因此动态建模就成为确定编码器压缩效率的关键。

下面将算术编码的 MATLAB 实现程序列举如下。

程序如下：

```
clear all
format long e;
symbol=['abcd'];
ps=[0.4 0.2 0.1 0.3];
inseq=('dacab');
codeword=suanshubianma(symbol,ps,inseq)
outseq = suanshujiema(symbol,ps,codeword,length(inseq))
%算术编码函数 suanshubianma
function acode=suanshubianma(symbol,ps,inseq)
high_range=[];
for k=1:length(ps)
     high_range=[high_range sum(ps(1:k))];
end
low_range=[0 high_range(1:length(ps-1))];
sbidx=zeros(size(inseq));
for i=1:length(inseq);
     sbidx(i)=find(symbol==inseq(i));
end
low=0;
higt=1;
for i=1:length(inseq);
     range=high-low;
     high=low+range*high_range(sbidx(i));
     low=low+range*low_range(sbidx(i));
end
acode=low;
%算术解码函数
function symbos=suanshujiema(symbol,ps,codeword,symlen)
format long e
high_range=[];
for k=1:length(ps)
     high_range=[high_range sum(ps(1:K))];
end
low_range=[0 high_range(1:length(ps)-1)];
psmin=min(ps);
symbos=[];
for i=1:symlen
     idx=max(find(low_range<=codeword));
     codeword=codeword-low_range(idx);
     if abs(codeword-ps(idx))<0.01*psmin
```



```
            idx=idx+1;
            codeword=0;
        end
        symbos=[symbos symbol(idx)];
        codeword=codeword/ps(idx);
        if abs(codeword)<0.01*psmin
            i=symlen+1;
        end
    end
```

运行结果为

```
    codeword=7.739200000000001e-001
    outseq=dacab
```

2.2.5 行程编码

计算机生成的图像往往有许多颜色相同的图片，在这些图片中，许多连续的扫描都具有同一种颜色，或者同一扫描行中有许多连续的像素都具有同样的颜色值。这种情况下，只需要存储一个像素及具有相同颜色的像素的数目即可，这种编码方法称为行程编码（Rum Length Encoding）。

把具有相同灰度值（颜色值）的一些像素序列称为一个行程。

对于简单的灰度图像，行程编码的数据结构表示见表 2-3。

表 2-3 行程编码数据结构

相同像素起始坐标	像素的灰度值
(k,j)	c

行程长度隐含在起始坐标中，不必单独列出。

下面通过一段 MATLAB 程序段，对行程编码的实现方法进行说明，希望能使读者对行程编码有更清楚的理解。在这里，将以如图 2-7 所示的原始图像为例进行分析。

程序代码示例如下：

```
    clear
    I=imread('pears.png');
    imshow(I);
    IY=im2bw(I,0.5);
    figure,imshow(IY)
    [m n]=size(IY);
    c=IY(1,1);E(1,1:3)=[1 1 c];
    t=1;
    for k=1:m
        for j=1:n
            if(not(and(k= =1,j= =1)))
                if(not(I(k,j)==c))
                    Enew(t,1:3)=[k j I(k,j)];
                    c=I(k,j);
```



```
                t=t+1;
            end
        end
    end
end
```

执行程序后效果如图 2-8 所示。

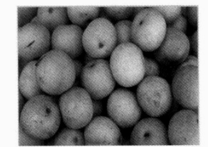


图 2-7　原始图像

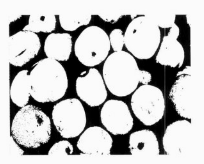


图 2-8　二值化图像

原图像 I=pears.png，大小为 IGRAY=486×732 字节=355752 字节，而行程编码的输出数组 E 为 197187×3 字节=591561 字节，比原来图像占用的存储空间还要大，可见压缩效果并不好。

如果首先对原始图像进行二值化，得到二值图像如图 2-8 所示。

运行同样的程序，将输入图像替换为二值化图像，则行程编码后的输出结果为 54700×3 字节=164100 字节，而二值化后的图像占用的存储空间依然为 486×732 字节=355752 字节，压缩比为 164100/355752=0.46，相对原始图像的压缩，压缩效果比较好。由此也说明，行程编码对于仅包含很少几个灰度级的图像，特别是二值图像，非常有效。特别地，该编码方法对有单一颜色背景下物体的图像，具有更高的压缩比。对于其他类型的图像压缩，其压缩比较低，甚至在最坏的情况下，如图像的每一个像素都与它周围的像素不同，行程编码（RLE）甚至可将文件的大小加倍，达不到编码压缩的目的。

因为这里是无失真编码，所以反编码后的图像与原图像是一样的，只能以编码后占用的存储空间来进行算法的比较，所以在上面只列出了编码前后图像占用空间的大小。

2.3　预测编码

预测编码（Predictive Coding）是指依据某一模型，根据以往的样本值对于新样本值进行预测，然后将样本的实际值与预测值相减得到一个误差值，对这一误差值进行编码。如果模型足够好且样本序列在时间和空间上存在较强的相关性，那么误差信号的幅度将远远小于信源原始信号，从而可以用较少的电平量对其差值量化得到较大的数据压缩结果。预测编码分为线性预测和非线性预测两类。

若能找到一个数学模型可以完全代表数据源，那么在接收端就能依据这一数学模型准确无误地产生出这些数据。但没有一个实际的系统能找到其完全准确的数学模型，因此，最好

的办法是采用预测器以某种最小化的误差方法对下一个样本进行预测。

1. DPCM的工作原理

在线性预测中，最常用的是差分脉冲编码调制，即DPCM（Differential Pulse Code Modulation）。Oliver在1952年对图像线性预测进行了理论研究，1958年，Graham用计算机进行了DPCM模拟。DPCM的工作原理如图2-9所示，主要是基于图像中相邻像素间的数据具有较强的相关性，每个像素可以根据以前已知的几个像素值进行预测。在DPCM编码中，编码和传输的不是像素值本身，而是这个取样值的预测值。

其中，x_N为t_N时刻的亮度取样值，$\hat{x}_N$为预测器根据t_N时刻之前的样本值$x_1,x_2,x_3,\cdots,x_{N-1}$对$x_N$所做的预测值，$e_N$为差值信号，即

$$e_N = x_N - \hat{x}_N \tag{2-13}$$

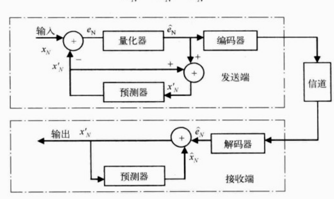


图2-9 DPCM系统原理框图

量化器对e_N进行量化得到$\hat{e}_N$，编码器对$\hat{e}_N$进行编码。接收端解码时的预测过程与发送端相同，所采用的预测器也相同。接收端恢复的输出信号x'_N是x_N的近似值，两者的误差为

$$\Delta x_N = x_N - (\hat{x}_N + \hat{e}_N) = x_N - x'_N = e_N - \hat{e}_N \tag{2-14}$$

当Δx_N足够小时，输入信号x_N和DPCM系统的输出信号x'_N接近一致。

2. 线性预测编码

在图像信源数据序列中，由$x_1,x_2,x_3,\cdots,x_{N-1}$对$x_N$进行预测。由于是对$x_N$进行线性预测，令$x_N$的预测值（估计值）为$\hat{x}_N$，则$\hat{x}_N$是$x_1,x_2,x_3,\cdots,x_{N-1}$的线性组合。

设二维图像信号$x(t)$是均值为零、方差为σ^2的平稳随机过程，$x(t)$在$t_1,t_2,t_3,\cdots,t_{N-1}$时刻的抽样值分别为$x_1,x_2,x_3,\cdots,x_{N-1}$，那么$x_N$时刻抽样的线性预测值为

$$\hat{x}_N = \sum_{i=1}^{N-1} a_i x_i = a_1x_1 + a_2x_2 + a_3x_3 + \cdots + a_{N-1}x_{N-1} \tag{2-15}$$

式中，a_i为预测系数，即待定常数。

若各a_i确定，则可以根据上式构成线性预测器。根据线性预测定义，$\hat{x}_N$应非常逼近x_N，这就要求各a_i为最佳系数。采用均方误差最小的准则，可求得各最佳的系数。

现定义x_N的均方误差为

$$E\{[e_N]^2\} = E\{[x_N - \hat{x}_N]^2\} \tag{2-16}$$

为使$E\{[e_N]\}$最小，对式（2-16）微分可得

$$\begin{aligned}\frac{\partial}{\partial a_i}E\{[e_N]^2\}&=\frac{\partial}{\partial a_i}E\{[x_N-\hat{x}_N]^2\}\\&=\frac{\partial}{\partial a_i}E\{[x_N-(a_1x_1+a_2x_2+a_3x_3+\cdots+a_{N-1}x_{N-1}]^2\}\\&=-2E\{[x_N-(a_1x_1+a_2x_2+a_3x_3+\cdots+a_{N-1}x_{N-1})x_i]\}\end{aligned}\tag{2-17}$$

式中，$i=1,2,3,\cdots,N-1$。

根据极值条件，可得如下$N-1$个线性方程：

$$\begin{cases}E\{[x_N-(a_1x_1+a_2x_2+a_3x_3+\cdots+a_{N-1}x_{N-1})x_1]\}=0\\E\{[x_N-(a_1x_1+a_2x_2+a_3x_3+\cdots+a_{N-1}x_{N-1})x_2]\}=0\\E\{[x_N-(a_1x_1+a_2x_2+a_3x_3+\cdots+a_{N-1}x_{N-1})x_3]\}=0\\\cdots\cdots\cdots\cdots\\E\{[x_N-(a_1x_1+a_2x_2+a_3x_3+\cdots+a_{N-1}x_{N-1})x_{N-1}]\}=0\end{cases}\tag{2-18}$$

该方程组可表示为

$$E\{[x_N-(a_1x_1+a_2x_2+a_3x_3+\cdots+a_{N-1}x_{N-1})x_i]\}=0$$

式中，$i=1,2,3,\cdots,N-1$。

令x_i和y_i的协方差为

$$R_{ij}=E(x_i,x_j)\qquad i,j=1,2,3,\cdots,N-1\tag{2-19}$$

则上式可以表示为

$$R_{Ni}=\sum_{k=0}^{N-1}a_{ki}R_{ki}=a_{1i}R_{1i}+a_{2i}R_{2i}+a_{3i}R_{3i}+\cdots+a_{(N-1)i}R_{(N-1)i}\tag{2-20}$$

若所有的协方差R_{ij}已知或可以测出时，则通过上式可计算出$N-1$个预测系数a_i。

综上所述，可以得出以下几点结论：

1）预测模型的复杂程度取决于线性预测中使用以前的样本数目，样本点越多，则预测器越复杂，最简单的预测仅使用前一个样本点，称为前值预测。

2）若采用x的同一行中x_N的若干已知像素样本值，如$x_1,x_2,x_3,\cdots,x_{N-1}$来对$x_N$进行预测，则称为一维预测。

3）若采用同一行及前几行内的已知像素样本值来预测x_N，则称为二维预测。

4）若采用的已知像素不仅是前几行的，而且还包括前几帧的，那么则称为三维预测。

图 2-10 所示是一个简单的、在 JPEG 无损编码方案中采用的实际预测器，它给出了静止图像的一个完整的二维预测器。它只考虑临近 3 点A、B和C的值，例如，以A、B或C作为Y的预测值，则线性预测方法见表 2-4。

表 2-4　Y 的预测值方法

预测方法	0	1	2	3	4	5	6	7
预测值　$\hat{y}$	非预测	A	B	C	$A+B-C$	$A+\frac{B-C}{2}$	$B+\frac{A-C}{2}$	$\frac{A+B}{2}$

B	A
C	Y

图 2-10　预测器的组成

3．非线性预测

线性预测编码的基础和前提是将整个图像域视为一个平稳随机过程，其自相关系数与像素在图像信源数据域中的位置无关。而实际上，图像的起伏变化是始终存在的，被描述的像素和周围像素之间存在多种多样的关系。线性预测系数是一种近似条件下的常数，它忽略了不同像素之间的个性条件，因此存在一些不足之处。

非线性预测针对线性预测的不足，充分考虑了图像的统计特性和图像信源数据的个别变化情况，即力求使预测系数与图像的实际局部特性相一致，通过使预测系数随预测条件变化，从而进一步提高压缩编码的性能。

2.4　图像的变换编码

变换编码主要包括 DFT 变换、K-L 变换、WHT 变换、DCT 变换和小波变换编码等，变换编码因其独特的编码效果，已经成为一种得到广泛应用的图像压缩编码方法。

1．变换编码的基本原理

图像变换编码的基本原理是将空域中描述的图像数据经过某种变换，如 DFT 变换、DCT 变换、K-L 变换等二维正交变换，转换到新的变换域中进行描述，在变换域中达到改变能量分布的目的，将图像能量在空间域的分散分布，变为在变换域中的相对集中分布，从而实现对信源图像数据的有效压缩。

变换编码的基本流程如图 2-11 所示，图像数据经过某种变换、量化和编码（通常为变长编码）后由信道传输到接收端，接收端进行相反的处理，即变长解码、反量化以及逆变换，然后输出原图像数据。

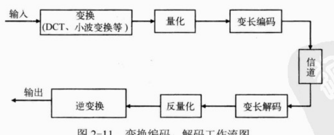


图 2-11　变换编码、解码工作流图

图像数据经过正交变换后，空域中的总能量在变换域中得到保持，但能量将会重新分布，并集中在变换域中少数的变换系数上，以达到压缩数据的目的。

2．正交变换的物理意义

通常情况下，变换编码都会选择正交变换，正交变换是一种数据处理手段，它将被处理的图像信源数据按照正交变换规则映射到另一个域进行处理。由于图像是以二维矩阵表示的，所以在图像编码中多采用二维正交变换形式。图像数据正交变换后不改变信源的熵值，

变换前后图像的信息量没有损失，完全可以通过对应的逆变换得到原来的图像数据。但统计分析表明，经过正交变换后，数据的分布规律发生了很大的改变，像素之间的相关性下降，变换系数向新坐标系中的少数坐标点集中，一般集中于少数的直流或低频分量的坐标点。变换编码将统计上高度相关的像素所构成的矩阵通过正交变换，变成统计上彼此较为独立，甚至达到完全独立的变换系数矩阵，以达到压缩数据的目的，这就是图像变换。

需要指出的是，如果将整个图像作为一个二维矩阵，则变换处理运算量太大，难以实现。所以在实际应用中，先将一幅图像分割成若干小的图像子块，如 8×8 或 16×16 小方块，各图像子块的像素值都可以被看成一个二维数据矩阵，变换是以这些图像子块为单位进行的。

3．变换类型与子块大小的选择

根据数字信号处理理论，所有正交变换中以 K-L 变换性能最优，经 K-L 变换后各变换系数在统计上不再相关，其协方差矩阵为对角矩阵，因而大大减小了图像原始数据的冗余度。因此，K-L 变换能完全消除图像子块内像素间的相关性，若舍弃一些特征值较小的变换系数，那么所引起的均方误差是所有正交变换中最小的。由于 K-L 变换是以原始图像各子块协方差矩阵的特征向量作为变换后的基向量，因此 K-L 变换的基对不同图像是不同的，与编码对象的统计特性有关，这种变换基的不确定性使得 K-L 变换在实际应用中不太方便。因此，尽管 K-L 变换具有许多明显的优点，但一般只用于进行理论上的比较。

DFT 变换是应用最早，且非常成熟的变换之一，性能接近于最佳，且具有快速算法，其不足之处在于图像子块的变换系数在边界处不连续而造成恢复后的图像子块边界也不连续（即存在 Gibbs 现象），于是由图像子块构成的整幅图像将呈现隐约可见的以图像子块为形状的小块结构，影响了图像质量，一定程度上影响了其应用。

DCT 变换是图像变换中应用得最多的变换编码，其性能接近于 K-L 变换，且变换矩阵与图像内容无关。根据 DCT 变换的特点，还可避免 DFT 变换中图像子块边界处产生的跳跃与 Gibbs 现象。此外，市场上拥有许多基于 DFT 快速算法的 ASIC 芯片，因此，DCT 变换已经成为图像变换编码的主流。目前，JPEG、MPEG、H.263 等国际编码标准选择采用 DCT 变换模块。

沃尔什变换与 DCT 变换相比，其算法简单，因而运算速度较快，适用于高速实时系统，而且实现该算法的硬件结构简单，不足之处是性能比 DCT 变换要差。

在确定变换方式之后，还需要选择变换块的大小。由于压缩的依据是基于子块内图像像素间的相关性，若子块选得太小，则不利于压缩比的提高。理论上，子块越大，计入的相关像素越多，压缩比就越大。但如果子块过大，则计算量太大，同时考虑到距离较远的像素间相关性并不高，实际上过大的子块对压缩比的提高效果反而不好。因此，图像变换中一般选择采用 8×8 或 16×16 大小的图像子块。

4．变换编码的实现步骤

根据图像变换编码的原理以及如图 2-11 所示的编码、解码逻辑流程，实现变换编码一般包含以下步骤。

（1）原始图像分块

根据编码的具体要求，将图像划分为若干 $N \times N$ 的图像子块，即

$$X = \begin{pmatrix} x_{00} & x_{01} & \cdots & x_{0(N-1)} \\ x_{10} & x_{11} & \cdots & x_{1(N-1)} \\ x_{20} & x_{21} & \cdots & x_{2(N-1)} \\ \vdots & \vdots & & \vdots \\ x_{(N-1)0} & x_{(N-1)1} & \cdots & x_{(N-1)(N-1)} \end{pmatrix} \tag{2-21}$$

通常情况下，N 取值为 8 或 16。图像分块之后，应同时根据编码的性能要求，综合考虑相关要素，选择变换矩阵 A 对各图像子块进行相应的正交变换。

设 Y 表示变换域中的图像数据，则可表示为

$$Y = AX \tag{2-22}$$

（2）变换域采样

根据一定的准则，对变换域中的系数进行合理的取舍。

（3）系数量化

由于变换之后的系数是不相关的，因此具有更大的独立性和有序性，利用量化使图像数据得到压缩。量化是产生有损压缩的原因，因此应选择合适的量化方法，以使量化失真最小。均方误差是衡量各种变换编码效能的一个重要准则。该准则可在较高的压缩比和一定的允许失真度之间寻求一个较理想的、可用的变换编码方式。

方均误差定义为

$$e = E\left[\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(y_{ij} - \hat{y}_{ij})^2\right] \tag{2-23}$$

式中，$\hat{y}_{ij}$ 为 y_{ij} 的量化值。

20 世纪 50 年代期间，Panter、Dire 和 Max 研究了使单个系数均方误差极小化的量化方案。研究发现，如果 y_{ij} 的概率密度函数是均匀的，那么具有均匀间隔输出的量化器是最佳的。对于其他的分布，使用非均匀量化器则能够起到减小均方误差的作用。

（4）解码与反变换

在变换编码系统的接收端对所接收的比特流进行解码，分离出各变换系数 $\hat{y}_{ij}$，并进行系数的舍入，被舍弃的系数均以 0 代替，并进行逆变换运算，恢复各图像子块及整幅图像。

2.5 数据压缩编码的国际标准

在多媒体应用中，视频数据量方面可压缩的信息量最多，而压缩处理后视频质量的高低则是决定多媒体服务质量好坏的主要因素，因此视频压缩技术是多媒体应用的核心技术。

在学术和应用领域，众多研究人员都致力于视频压缩技术的研究，并且制定了几个标准，如 JPEG、MPEG 和 H.261 标准等。这些标准覆盖了较大的视频范围和应用领域，支持不同速率、不同图像质量要求、不同复杂度、容错性和实时性的视频业务，能够满足包括电视会议、视频电子邮件、可视电话、广播级视频应用等不同要求的服务。随着视频应用需求的不断发展，视频压缩技术也有了很大的提高，新出现的压缩标准有了更高的压缩效率（在相同图像质量下需要更低的传送码率或在相同传输速率下提供质量更好的图像），同时支持各种传输速率以适应不同的传送网络需求。

下面将对 JPEG 标准和 MPEG 标准进行介绍。

2.5.1 JPEG 标准

JPEG（Joint Photographic Experts Group）是一个由 ISO 和 IEC 两个组织机构联合组成的专家组，负责制定静态和数字图像数据压缩编码标准，这个专家组开发的算法称为 JPEG 算法，并且成为国际上通用的标准，因此又称为 JPEG 标准。JPEG 是一个适用范围很广的静态图像数据压缩标准，既可用于灰度图像压缩又可用于彩色图像压缩。JPEG 专家组开发了两种基本的压缩算法，一种是以离散余弦变换 DCT 为基础的有损压缩算法；另一种是以预测技术为基础的无损压缩算法。使用有损压缩算法时，在压缩比为 25∶1 的情况下，压缩后还原得到的图像与原始图像相比较，对于非图像专家而言，很难找出它们之间的区别。因此，该压缩技术得到了广泛的应用。为了在保证图像质量的前提下进一步提高压缩比，近年来，JPEG 专家组又制定了 JPEG 2000（简称 JP 2000）标准，JPEG 2000 与传统 JPEG 最大的不同在于：它放弃了传统 JPEG 所采用的以离散余弦变换为主的区块编码方式，而改用以小波变换为主的多解析编码方式。采用小波变换的主要目的是为了将图像的频率成分抽取出来。

图 2-12 为 JPEG 标准的基本处理框图。

JPEG 标准将整个图像分成 8×8 的图像子块，并作为二维离散余弦变换 DCT 的输入。通过 DCT 变换，把能量集中在少数几个系数上，然后对这些系数进行量化。由于人眼对亮度信号比对色差信号更敏感，因此 JPEG 使用了两种量化表：亮度量化值和色差量化值。此外，由于人眼对低频分量的图像比对高频分量的图像更敏感，所以对图中左上角的量化步长要比右下角的量化步长小。

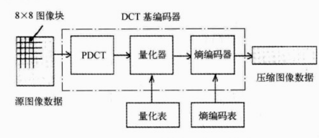


图 2-12　JPEG 标准的基本处理框图

在经过量化后，就是进行熵编码过程，将 DC 系数进行 DPCM 编码，将 AC 系数按 Z 形排列之后采用 RLE 编码，最后得到经压缩编码后的数据值。

此外，JPEG 还规定了 4 种运行模式，以满足不同的应用需要。

1）基于 DPCM 的无损编码模式：压缩比可以达到 2∶1。

2）基于 DCT 的有损顺序编码模式：压缩比可以达到 10∶1 以上。

3）基于 DCT 的递增编码模式。

4）基于 DCT 的分层编码模式。

下面，以 JPEG 有损顺序编码算法为例说明如何利用 JPEG 标准进行图像压缩。其主要的计算步骤如下。

1）将源图像分成几个颜色平面（分量图像）。

2）对每个 8×8 数据块进行正向离散余弦变换（FDCT）。

3）根据表 2-5 和表 2-6 中的理化模板系数对 DCT 系数进行量化。

4）Z 形排列量化结果。

5）使用差分脉冲编码调制（DPCM）对直流系数（DC）进行编码。

6）使用行程长度编码（RLE）对交流系数（AC）进行编码。

7）熵编码（Entropy Coding）。



表 2-5 亮度的量化模板系数

16	11	10	16	24	40	51	66
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

表 2-6 颜色的量化模板系数

17	18	24	67	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

针对以上处理步骤，以下的 MATLAB 程序代码示例说明了 JPEG 有损顺序编码算法的实现过程。

```
%JPEGdemo.m JPEG 算法仿真
%本程序只是为了说明基本的 JPEG 功能
clear all
%待处理数据
f=[52 55 61 66 70 61 64 73
   63 59 66 90 109 85 69 72
   62 59 68 113 144 104 66 73
   63 58 71 122 154 106 70 69
   67 61 68 104 126 88 68 70
   79 65 60 70 77 68 58 75
   85 71 64 59 55 61 65 83
   87 78 69 68 65 76 78 94];
%f 也可以是读入灰度图像数据
echo on
```



```
% level shift by 128
f=f-128;
[mf,nf]=size(f);mb=mf/8;nb=nf/8;
%计算 f 的大小，以及分块后的块数
%步骤一：计算每个 8×8 图像子块的离散 DCT 系数
Ff=blkproc(f,[8 8],'dct');
%对 f 的每个图像子块的列进行 DCT 变换 apply DCT to each column of each block of f
Ff=blkproc(Ff',[8 8],'dct');
%对 f 的每个图像子块的行进行 DCT 变换 apply DCT to each row of each block of f
Ff=round(Ff');
%按比例量化
Q=[16 11 10 16 24 40 51 61
   12 12 14 19 26 58 60 55
   14 13 16 24 40 57 69 56
   14 17 22 29 51 87 80 62
   18 22 37 56 68 109 103 77
   24 35 55 64 81 104 113 92
   49 64 78 87 103 121 120 101
   72 92 95 98 112 100 103 99];
%量化矩阵
Fq=round(blkproc(Ff,[8 8], 'divq', Q));%取整
echo off
%对 DC 系数进行 DPCM 编码，逐行扫描
if mb*nb>1,
     fdc=reshape(Fq(1:8:mf,1:8:nf)',mb*nb,1);
     fdpcm=dpcm(fdc,1);
else
     fdpcm=Fq(1,1)-(-17);
end
dccof=[];
for i=1:mb*nb,
     dccof=[dccof jdcenc(fdpcm(i))];
end
disp(['Differential DC coefficient(num2str(fdpcm))is encoded as:']);
disp(int2str(dccof));
echo on
%之字形扫描 AC 系数，进行行程（LZW）编码
z=[1 2 6 7 15 16 28 29
   3 5 8 14 17 27 30 43
   4 9 13 18 26 31 42 44
   10 12 19 25 32 41 45 54
   11 20 24 33 40 46 53 55
   21 23 34 39 47 52 56 61
   22 35 38 48 51 57 60 62
   36 37 49 50 58 59 63 64];
echo off
```

```
acseq=[];
for i=1:mb
    for j=1:nb
        tmp(z)=Fq(8*(i-1)+1:8*i,8*(j-1)+1:8*j);
        %tmp 为 1/64
        eobi=max(find(tmp~=0));
        acseq=[acseq tmp(2:eobi) 999];%给 eob 赋值 999
    end
end
accof=jacenc(acseq);
disp(['DC coefficient after Hoffman coding has'int2str(length(dccof))...
    ' bits']);
disp(['AC coefficient after Hoffman coding has'int2str(length(accof))...
    ' bits']);
```

得到结果：

```
DC coefficient after Hoffman coding has 7 bits
AC coefficient after Hoffman coding has 84 bits
```



2.5.2 MPEG 视频编码压缩标准

从时间的观点看，数字图像分为静态图像和运动图像，视频信号就是典型的运动图像。视频压缩的目标是在尽可能保证视觉效果的前提下减少视频数据率。视频压缩比一般指压缩后的数据量与压缩前的数据量之比。

根据压缩前和解压缩后的数据是否完全一致，视频压缩可分为有损压缩和无损压缩。无损压缩意味着解压缩后的数据与压缩前的数据完全一致。有损压缩则意味着解压缩后的数据与压缩前的数据不一致。在压缩的过程中要丢失一些人眼和人耳所不敏感的图像或音频信息，而且丢失的信息不可恢复。丢失的数据率与压缩比有关，压缩比越小，丢失的数据越多，解压缩后的效果越差。此外，某些有损压缩算法采用多次重复压缩的方式，这样还会引起额外的数据丢失。

具体的视频编码和解码过程如图 2-13 所示。

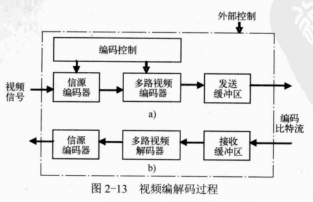


图 2-13 视频编解码过程

a) 视频编码 b) 视频解码

图 2-14 所示为视频信号的压缩过程，该图充分说明了视频信号的压缩包括两个主要方面：帧内压缩与帧间压缩。帧内压缩（Intraframe Compression）也称为空间压缩（Spatial Compression），当压缩一帧图像时，仅考虑本帧的数据而不考虑相邻帧之间的冗余信息，这实际上与静态图像压缩类似。帧内压缩一般达不到很高的压缩。帧间压缩（Interframe Compression）是基于许多视频或动画的连续前后两帧具有的相关性，或者说前后两帧信息变化很小的特点，即连续的视频其相邻帧之间具有冗余信息。根据这一特性，压缩相邻帧之间的冗余量就可以进一步提高压缩量，减小压缩比。帧间压缩也称为时间压缩（Temporal Compression），它通过比较时间轴上不同帧之间的数据进行压缩。帧间压缩一般是无损压缩。

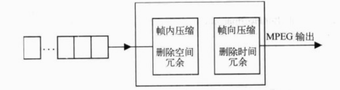


图 2-14　视频信号的压缩过程

MPEG（Moving Picture Experts Group）是运动图像专家组的简称，该组织专门致力于对数字存储媒介中运动图像及其伴音的压缩编码技术的标准化和实用化研究。MPEG 是个国际标准，即 ISO 11172。该小组于 1991 年底提出了用于数字存储媒介的、速率约为 1.5MB/s 的运动图像及其伴音的压缩编码，并于 1992 年正式通过，通常被称为 MPEG 标准，此标准后来被定名为 MPEG-1。MPEG 与 JPEG 算法在概念上类似，只不过它还利用了相继图像之间的冗余信息。由于可达到 100∶1 的压缩比，所以 MPEG 算法非常实用，可用于在 1Mbit/s 的信道中传送带声音的彩色电视图像，以及在磁盘驱动器中存储较长一段时间的数字电视图像片段等。

到目前为止，MPEG 标准已不再是一个单一的标准，而是一个用于全运动视频和相关音频压缩的标准系列，包括 MPEG-1、MPEG-2、MPEG-3、MPEG-4 和 MPEG-7 共 5 个标准，每一个标准都有其特定的应用范围。其中，它的两个标准——MPEG-1 和 MPEG-2 标准的应用范围最广，也特别重要。MPEG-1 用于加速 CD-ROM 中图像的传输。它的目的是把 221MB/s 的 NTSC 图像压缩到 1.2MB/s，压缩率为 200∶1。这是图像压缩的工业认可标准。MPEG-2 用于图像的宽带传输，图像质量达到电视广播甚至 HDTV 的标准。和 MPEG-1 相比，MPEG-2 支持更广的分辨率比特率范围，将成为数字图像光盘（DVD）和数字广播电视的压缩方式。这些市场和计算机市场交织在一起，从而使 MPEG-2 成为计算机的一种重要的图像压缩标准。这一点非常重要，因为将 MPEG-1 的比特流解压缩时需要用到 MPEG-2 的解压缩器。MPEG-4 标准支持非常低的比特率数据流的应用，如电视电话，视频邮件和电子报刊等。

MPEG 视频压缩分为空间域压缩与时间域压缩。MPEG 标准在空间域的压缩，类似于 JPEG 标准。每一帧被作为独立的图像获取，且压缩步骤与 JPEG 标准的步骤一样。时间域压缩，即帧间编码的基本思想是仅存储运动图像从一帧到下一帧的变化部分，而不是存储全部图像数据，这样做能极大地减少运动图像数据存储量，达到帧间压缩的目的。这是通

过把帧序列划分成I帧、P帧、B帧，使用参照帧及运动补偿技术来实现的。所谓的I帧是在解码时，无需参照任何其他帧的帧，或称为内编码帧，它是利用自身的相关性进行帧内压缩编码；而在帧编码时仅使用最近的前一帧（I 帧或 P 帧）作为参照帧时，该帧称为 P 帧或称为预测帧；对于在帧编码时要使用前、后帧作为参照帧时，该帧称为 B 帧，或称为双向预测帧。



2.6 小结

本章在分析图像编码的必要性与可能性的基础上，对图像编码与压缩的概念、理论进行了简要介绍，并从无损压缩和有损压缩的角度具体介绍了几种常用的图像编码与压缩技术。

有损编码是以丢失部分信息为代价来换取高压缩比的。有损压缩方法主要有有损预测编码方法、变换编码方法等。预测编码是根据某一模型利用以往的样本值对于新样本值进行预测，然后将样本的实际值与其预测值相减得到一个误差值，对于这一误差值进行编码。如果对差值信号不进行量化而直接编码，称为无损预测编码。如果不是直接对差值信号进行编码，而是对差值信号进行量化后再进行编码，称为有损预测编码。有损预测方法有多种，其中差分脉冲编码调制是一种具有代表性的编码方法。由于离散余弦变换可与最佳变换 K-L 变换媲美，而计算复杂度适中，近年来在图像数据压缩中，采用离散余弦变换编码的方案很多。JPEG、MPEG、H.261 等压缩标准，都用到离散余弦变换编码进行数据压缩。

JPEG 是联合图像专家小组开发研制的连续色调、多级灰度、静止图像的数字图像压缩编码方法。JPEG 中的核心算法是 DCT 变换编码。JPEG 采用的是 8×8 图像子块的二维离散余弦变换。在编码器的输入端，把原始图像顺序地分割成一系列 8×8 图像子块的 64 个变换系数经量化后，按直流系数 DC 和交流系数 AC 分成两类处理。JPEG 对 DC 系数采用 DPCM 编码，即对相邻块之间的 DC 系数的差值编码。其余 63 个交流系数采用行程编码。为了进一步达到压缩数据的目的，可以对 DPCM 编码后的 DC 码和 RLE 编码后的 AC 码的码字再作熵编码。JPEG 建议使用两种熵编码方法：哈夫曼（Huffman）编码和自适应二进制算术编码。

MPEG 视频压缩分为空间域压缩与时间域压缩。MPEG 标准在空间域的压缩，类似于 JPEG 标准。每一帧被作为独立的图像获取，且压缩步骤与 JPEG 标准的步骤一样。时间域压缩，即帧间编码的基本思想是仅存储运动图像从一帧到下一帧的变化部分，而不是存储全部图像数据，这样做能极大地减少运动图像数据的存储量，达到帧间压缩的目的。它通过把帧序列划分成I帧、P帧、B帧，使用参照帧及运动补偿技术来实现。

习题

2-1 阐述图像编码的两个评价准则。

2-2 DCT 系数量化后，为什么需要进行 Z 形扫描？

2-3 设有一幅图像的 8×8 图像子块的亮度数据和 DCT 变换系数见表 2-7 和表 2-8，试对其按 JPEG 基本系统进行编码。

表 2-7　原始图像 8×8 子块亮度数据表

117	120	109	77	73	64	54	60
139	123	102	74	75	60	64	87
109	100	93	85	70	68	97	403
97	117	117	78	74	94	103	79
164	149	88	87	99	91	74	68
147	94	90	102	84	72	82	102
95	92	116	119	114	122	137	150
111	112	140	150	157	163	161	157

表 2-8　DCT 变换系数表

102	7	6	0	0	0	0	0
−15	11	4	0	−1	−2	0	0
6	−5	−3	−2	0	0	0	0
−6	8	2	2	3	0	0	0
4	2	−3	2	−3	−1	0	0
−2	−5	−1	−3	−3	1	0	0
−1	0	0	0	2	1	0	0
1	0	0	0	0	−1	1	0

2-4　已知符号 a、e、i、o、u、k 的出现概率分别是 0.2、0.3、0.1、0.2、0.1、0.1，请对 0.23355 进行算术解码。

2-5　简述变换编码的原理及过程。

2-6　说明 MPEG 中 I，P，B 帧的含义。

2-7　已知信源 $X\{0, 1\}$，信源符号概率为 P(0)=1/4，P(1)=3/4。试对 1001 和 10111 进行算术编码。







第3章 图像复原

图像复原是图像处理的另一个重要内容，它的主要目的是改善给定的图像质量并尽可能恢复原图像。图像在形成、传输和记录过程中，受多种因素的影响，图像的质量都会有所下降，典型表现有图像模糊、失真、有噪声等。这一质量下降的过程称为图像的退化。图像复原（或称图像恢复）的目的就是尽可能恢复退化图像的本来面目。本章主要讲解一些基本的图像复原技术。

3.1 图像复原的基本概念

无论是由光学、光电或电子方法获得的图像都会有不同程度的退化。由于获得图像的方法不同，其退化形式是多种多样的，如传感器噪声、摄像机未聚焦、物体与摄像设备之间的相对移动、随机大气湍流、光学系统的像差、成像光源或射线的散射、摄影胶片的非线性和几何畸变等，这些因素都会使成像的分辨率和对比度退化。如果对退化的类型、机制和过程都十分清楚，就可以利用其反过程把已退化的图像复原。图像复原结果的好坏主要取决于对图像退化过程的先验知识掌握的精确程度。

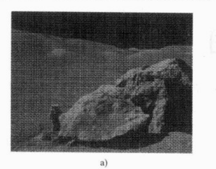
对图像复原结果的评价有一些准则，这些准则包括最小方均误差准则、加权均方准则，最大熵准则等。这些准则是规定复原后的图像与原图像相比较的质量标准，也就是说，当确定图像复原的质量标准后，对所期望的结果做出符合某种标准的最佳估计。典型的图像复原是根据图像退化的先验知识建立一个退化模型，以此模型为基础，采用各种反退化处理方法，如滤波等，使图像复原后符合某些准则，图像质量得到改善。

从某种意义上说，图像复原和图像增强的目的都是为了改善图像的质量，但它们的技术思想却完全不同，二者之间有着很大的区别。图像增强不考虑图像是如何退化的，不建立或很少建立模型，只通过各种技术来增强图像的视觉效果，以适应人视觉系统的生理、心理特点，从而使人觉得舒适、愉悦，却很少涉及客观和统一的评价标准。因此，图像增强可以不顾及增强后的图像是否符合原图像、是否失真，往往只要看着舒适即可。图像复原就完全不同，需要知道图像退化机制和过程的先验知识，要建立相应的退化模型，据此找出一种相应的反过程，从而恢复出原图像。例如，未聚焦的照片，无论用什么增强方法也不可能得到清晰的原图像，但是，若已知其退化的先验知识是镜头不聚焦，则其反过程可用一阶贝塞尔函数的反滤波来复原图像。另外，图像复原要明确规定质量标准，以便对希望的结果做出最佳的估计。

由于图像复原过程的特殊性，可以根据不同的退化模型和不同的质量评价标准，推导出多种复原的方法。

下面给出两个图像复原的实例，以增加对图像复原技术的直观印象。图 3-1 所示是巴特沃斯带阻滤波器复原受正弦噪声干扰的图像的例子。可以看到，恢复后的图像效果非常好，即使是细小的细节和纹理都被这种简单的滤波方式有效地修复了。

图 3-2 所示是一个维纳滤波器应用的例子，由于受剧烈大气湍流的严重影响，获得的图像已经模糊不清，通过维纳滤波器恢复出来的图像相当清晰，非常接近原始图像。


a)

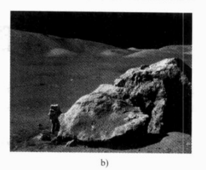

b)

图 3-1 用巴特沃斯带阻滤波器复原受正弦噪声干扰的图像

a) 被正弦噪声干扰的图像 b) 滤波效果图


a)

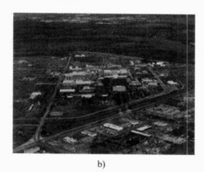

b)

图 3-2 维纳滤波器应用

a) 受大气湍流严重影响的图像 b) 用维纳滤器恢复出来的图像

3.2 图像退化模型

图像复原处理的关键是建立退化模型，原图像 $f(x,y)$ 是通过一个系统 H 及加入一个外来加性噪声 $n(x,y)$ 而退化成一幅图像 $g(x,y)$ 的，如图 3-3 所示。

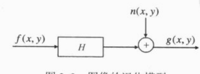


图 3-3 图像的退化模型

图像复原可以被看成一个预测估计的过程，由已给出的退化图像 $g(x,y)$ 估计出系统参数 H，从而近似地恢复出 $f(x,y)$。$n(x,y)$ 为一种统计性质的信息。为了对处理结果做出某种最佳估计，一般还应首先确立一个质量标准。复原处理的基础在于对系统 H 的基础了解，系统是由某些元件或部件以某种方式构造而成的整体。系统本身所具有的某些特性就构成了

系统的输入信号与输出信号的某种联系，这种联系从数学上可以用算子或响应函数 $h(x,y)$ 来描述。

因此图像退化过程的数学表达式就可以写为

$$g(x,y)=H[f(x,y)]+n(x,y) \tag{3-1}$$

$H[\cdot]$可理解为综合所有退化因素的函数或算子。

抽象地讲，在不考虑加性噪声 $n(x,y)$ 时，图像退化的过程也可以被看成是一个变换 H，即

$$H[f(x,y)]\to g(x,y)$$

由 $g(x,y)$ 求得 $f(x,y)$，就是寻求逆变换 H^{-1}，使得 $H^{-1}[g(x,y)]\to f(x,y)$

图像复原的过程，就是根据退化模型及原图像的某些知识，设计一个恢复系统 $p(x,y)$，以退化图像 $g(x,y)$ 作为输入，该系统应使输出的恢复图像 $\hat{f}(x,y)$，按某种准则最接近原图像 $f(x,y)$，图像退化与复原的过程如图 3-4 所示。其中 $h(x,y)$ 和 $p(x,y)$ 分别称为成像系统和恢复系统的冲激响应。

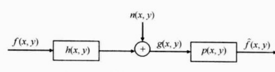


图 3-4 图像退化与复原的过程

系统 H 的分类方法很多，可分为线性系统和非线性系统；时变系统和非时变系统；集中参数系统和分布参数系统；连续系统和离散系统等。

线性系统就是具有均匀性和相加性的系统。当不考虑加性噪声 $n(x,y)$ 时，即令 $n(x,y)=0$，则图 3-3 所示的系统可表示为

$$g(x,y)=H[f(x,y)]$$

两个输入信号 $f_1(x,y)$、$f_2(x,y)$ 对应的输出信号为 $g_1(x,y)$、$g_2(x,y)$，如果有

$$H[k_1f_1(x,y)+k_2f_2(x,y)]=H[k_1f_1(x,y)]+H[k_2f_2(x,y)]=k_1g_1(x,y)+k_2g_2(x,y) \tag{3-2}$$

成立，则系统 H 是一个线性系统，k_1 和 k_2 为常数。

线性系统的这种特性为求解多个激励情况下的输出响应带来很大方便。

如果一个系统的参数随时间变化，即称为时不变系统或非时变系统；否则，该系统为时变系统。与此相对应，对二维函数来说，如果

$$H[f(x-\alpha,y-\beta)]=g(x-\alpha,y-\beta) \tag{3-3}$$

则 H 是空间不变系统（或称位置不变系统）。式中，α、β 分别是空间位置的位移量，表示图像中任一点通过该系统的响应只取决于在该点的输入值，而与该点的位置无关。

由上式可见，如果系统 H 有式（3-2）和式（3-3）的关系，那么系统 H 就是线性和空间位置不变的系统。在图像复原处理中，非线性和空间变化的系统模型虽然更具有普遍性和准确性，但它却给处理工作带来了巨大的困难，它常常没有解或很难用计算机来处理。实际的成像系统在一定条件下往往可以近似地视为线性和空间不变的系统，因此在图像复原处理中，往往用线性和空间不变的系统模型加以近似。这种近似使线性系统理论中的许多知识可以直接用于解决图像复原问题，所有图像复原处理特别是数字图像复原处理主要采用线性的空间不变复原技术。

3.2.1 连续的退化模型

单位冲激函数 $\delta(t)$ 是一个振幅在原点之外所有时刻为零，而在原点处振幅为无穷大，宽度无限小，面积为 1 的窄脉冲，其域表达式为

$$\delta(t)\begin{cases}\infty & t=0\\ 0 & t\neq 0\end{cases} \qquad \int_{-\infty}^{+\infty}\delta(t)\mathrm{d}t=1 \tag{3-4}$$

$\delta(t)$ 的卷积取样公式为

$$f(x)=\int_{-\infty}^{+\infty}f(x-t)\delta(t)\mathrm{d}t \tag{3-5}$$

或

$$f(x)=\int_{-\infty}^{+\infty}f(x)\delta(x-t)\mathrm{d}t \tag{3-6}$$

上述的一维时域冲激函数 $\delta(t)$ 可推广到二维空间域中，从而可把 $f(x,y)$ 写成下列形式：

$$f(x,y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(\alpha,\beta)\delta(x-\alpha,y-\beta)\mathrm{d}\alpha\mathrm{d}\beta \tag{3-7}$$

由于 $g(x,y)=H[f(x,y)]+n(x,y)$，如果令 $n(x,y)=0$，同时考虑到 H 为线性算子，则

$$\begin{aligned}g(x,y)&=H[f(x,y)]\\&=H\left[\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(\alpha,\beta)\delta(x-\alpha,y-\beta)\mathrm{d}\alpha\mathrm{d}\beta\right]\\&=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}H[f(\alpha,\beta)\delta(x-\alpha,y-\beta)]\mathrm{d}\alpha\mathrm{d}\beta\\&=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(\alpha,\beta)H[\delta(x-\alpha,y-\beta)]\mathrm{d}\alpha\mathrm{d}\beta\end{aligned} \tag{3-8}$$

令 $h(x,\alpha,y,\beta)=H[\delta(x-\alpha,y-\beta)]$，则有

$$g(x,y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(\alpha,\beta)h(x,\alpha,y,\beta)\mathrm{d}\alpha\mathrm{d}\beta \tag{3-9}$$

式中，$h(x,\alpha,y,\beta)$ 为系统 H 的冲激响应，即 $h(x,\alpha,y,\beta)$ 是系统 H 对坐标为 (α,β) 处的冲激函数 $\delta(x,\alpha,y,\beta)$ 的响应。在光学中冲激为一光点，因此又将 $h(x,\alpha,y,\beta)$ 称为退化过程的点扩散函数（PSF）。

式（3-9）说明：当系统 H 对冲激函数的响应为已知，则对任意输入 $f(x,y)$ 的响应均可由式（3-9）求得。也就是说，线性系统 H 完全可由其冲激响应来表征。

当系统 H 空间位置不变时，则

$$h(x-\alpha,y-\beta)=H[\delta(x-\alpha,y-\beta)] \tag{3-10}$$

这样就有

$$g(x,y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(\alpha,\beta)h(x-\alpha,y-\beta)\mathrm{d}\alpha\mathrm{d}\beta \tag{3-11}$$

即系统 H 对输入 $f(x,y)$ 的响应就是系统输入信号 $f(x,y)$ 与系统冲激响应的卷积。

考虑加性噪声 $n(x,y)$ 时，式（3-9）可写成

$$g(x,y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(\alpha,\beta)h(x,\alpha,y,\beta)\mathrm{d}\alpha\mathrm{d}\beta+n(x,y) \tag{3-12}$$

式中，$n(x,y)$ 与图像中的位置无关。

3.2.2 离散的退化模型

在连续的退化模型中，把 $f(\alpha,\beta)$ 和 $h(x-\alpha,y-\beta)$ 进行均匀取样后就可引伸出离散的退化模型。为了更好地理解离散的退化模型，我们首先用一维函数来说明其基本概念，然后再推广到二维。

设有两个函数 $f(x)$ 和 $h(x)$，它们被均匀取样后分别形成长度为 A 和 B 的一维矩阵，于是 $f(x)$ 变成在 $x=0,1,2,\cdots,A-1$ 范围内的离散变量，$h(x)$ 变成在 $x=0,1,2,\cdots,B-1$ 范围内的离散变量；于是 $f(x)$ 和 $h(x)$ 的连续卷积关系就变成离散卷积关系。

若 $f(x)$ 和 $h(x)$ 都是周期为 N 的序列，那么，它们的时域离散卷积定义为

$$g(x)=\sum_{m=0}^{N} f(m)h(x-m) \qquad x=0,1,2,\cdots,N \tag{3-13}$$

则 $g(x)$ 也是周期为 N 的序列，周期卷积可以用常规卷积法计算，也可用卷积定理进行快速卷积计算。

若 $f(x)$ 和 $h(x)$ 均为非周期性的序列，则可用延拓的方法延拓为周期序列，为避免折叠现象，可令周期 $M \geqslant A+B-1$，延拓后的 $f(x)$ 和 $h(x)$ 表示为

$$f_e(x)\begin{cases} f(x) & 0\leqslant x\leqslant A-1 \\ 0 & A-1<x\leqslant M-1 \end{cases} \tag{3-14}$$

$$h_e(x)\begin{cases} h(x) & 0\leqslant x\leqslant B-1 \\ 0 & B-1<x\leqslant M-1 \end{cases} \tag{3-15}$$

可得到离散卷积退化模型

$$g_e(x)=\sum_{m=0}^{M-1} f_e(m)h_e(x-m) \qquad x=0,1,2,\cdots,M-1 \tag{3-16}$$

因为 $f_e(x)$ 和 $h_e(x)$ 的周期为 M，所以 $g_e(x)$ 的周期也为 M。经过这样的延拓处理，一个非周期的卷积问题就变成了周期卷积问题了，因此可以用快速卷积法进行运算。

用矩阵形式表述离散的退化模型，可写成

$$\boldsymbol{g}=\boldsymbol{Hf} \tag{3-17}$$

式中

$$\boldsymbol{f}=\begin{pmatrix} f_e(0) \\ f_e(1) \\ \vdots \\ f_e(M-1) \end{pmatrix} \qquad \boldsymbol{g}=\begin{pmatrix} g_e(0) \\ g_e(1) \\ \vdots \\ g_e(M-1) \end{pmatrix}$$

$\boldsymbol{H}$ 是 $M\times M$ 阶矩阵

$$\boldsymbol{H}=\begin{pmatrix} h_e(0) & h_e(-1) & h_e(-2) & \cdots & h_e(-M+1) \\ h_e(1) & h_e(0) & h_e(-1) & \cdots & h_e(-M+2) \\ h_e(2) & h_e(1) & h_e(0) & \cdots & h_e(-M+3) \\ \vdots & \vdots & \vdots & & \vdots \\ h_e(M-1) & h_e(M-2) & h_e(M-3) & \cdots & h_e(0) \end{pmatrix} \tag{3-18}$$

利用 $h_e(x)$ 的周期性，$h_e(x)=h_e(\pm M+x)$，式（3-18）可写成：

$$H = \begin{pmatrix} h_e(0) & h_e(M-1) & h_e(-2) & \cdots & h_e(1) \\ h_e(1) & h_e(0) & h_e(-1) & \cdots & h_e(2) \\ h_e(2) & h_e(1) & h_e(0) & \cdots & h_e(3) \\ \vdots & \vdots & \vdots & & \vdots \\ h_e(M-1) & h_e(M-2) & h_e(M-3) & \cdots & h_e(0) \end{pmatrix} \tag{3-19}$$

可以看出，$\boldsymbol{H}$ 为一个循环矩阵，即每行最后一项等于下一行的第一项，最后一行的最后一项等于第一行的第一项。

从上述一维模型可以推广到二维情况。如果给出 $A\times B$ 大小的数字图像，以及 $C\times D$ 大小的点扩散函数，可首先做成大小为 $M\times N$ 的周期延拓图像。

$$f_e(x,y)=\begin{cases} f(x,y) & 0\leqslant x\leqslant A-1 \text{且} 0\leqslant y\leqslant B-1 \\ 0 & A-1<x\leqslant M-1 \text{或} B-1<y\leqslant N-1 \end{cases} \tag{3-20}$$

$$h_e(x,y)=\begin{cases} h(x,y) & 0\leqslant x\leqslant C-1 \text{且} 0\leqslant y\leqslant D-1 \\ 0 & C-1<x\leqslant M-1 \text{或} D-1<y\leqslant N-1 \end{cases} \tag{3-21}$$

为避免折叠，要求 $M\geqslant A+C-1$，$N\geqslant B+D-1$。这样一来，$f_e(x,y)$ 和 $h_e(x,y)$ 分别成为二维周期函数，它们在 x 和 y 方向上的周期分别为 M 和 N。由此得到的二维退化模型为一个二维卷形式

$$g_e(x,y)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f_e(m,n)h_e(x-m,y-n)+n_e(x,y) \tag{3-22}$$

式中，$x=0,1,2,\cdots,M-1$；$y=0,1,2,\cdots,N-1$。$g_e(x,y)$ 也为周期函数，其周期与 $f_e(x,y)$ 和 $h_e(x,y)$ 的周期完全一样。

上式也可用矩阵表示为

$$\boldsymbol{g}=\boldsymbol{Hf}+\boldsymbol{n} \tag{3-23}$$

式中，$\boldsymbol{g}$、$\boldsymbol{f}$、$\boldsymbol{n}$ 皆用行向量堆叠成 $M\times N$ 维，它把各行顺时针转 90°堆叠而成，都是 $M\times N$ 维列向量；$\boldsymbol{H}$ 为 $MN\times MN$ 的矩阵

$$\boldsymbol{H}=\begin{pmatrix} \boldsymbol{H}_0 & \boldsymbol{H}_{M-1} & \boldsymbol{H}_{M-2} & \cdots & \boldsymbol{H}_1 \\ \boldsymbol{H}_1 & \boldsymbol{H}_0 & \boldsymbol{H}_{M-1} & \cdots & \boldsymbol{H}_2 \\ \boldsymbol{H}_2 & \boldsymbol{H}_1 & \boldsymbol{H}_0 & \cdots & \boldsymbol{H}_3 \\ \vdots & \vdots & \vdots & & \vdots \\ \boldsymbol{H}_{M-1} & \boldsymbol{H}_{M-2} & \boldsymbol{H}_{M-3} & \cdots & \boldsymbol{H}_0 \end{pmatrix} \tag{3-24}$$

式中，每个 $\boldsymbol{H}_j$ 都是一个 $N\times N$ 的矩阵，是由延拓函数 $h_e(x,y)$ 的 j 行构成的。

$$\boldsymbol{H}_j=\begin{pmatrix} h_e(j,0) & h_e(j,N-1) & h_e(j,N-2) & \cdots & h_e(j,1) \\ h_e(j,1) & h_e(j,0) & h_e(j,N-1) & \cdots & h_e(j,2) \\ h_e(j,2) & h_e(j,1) & h_e(j,0) & \cdots & h_e(j,3) \\ \vdots & \vdots & \vdots & & \vdots \\ h_e(j,N-1) & h_e(j,N-2) & h_e(j,N-3) & \cdots & h_e(j,0) \end{pmatrix} \tag{3-25}$$

可见，$\boldsymbol{H}_j$ 是一个循环矩阵，而 $\boldsymbol{H}$ 是一个分块循环矩阵。

上述离散的退化模型是在线性空间不变的前提下推出的。目的是在给定了 $g(x,y)$，并且

知道 $h(x,y)$ 和 $n(x,y)$ 的情况下，估计出理想的原始图像 $f(x,y)$ 。但是，要想从式（3-23）直接求解得 $f(x,y)$ ，对于实际大小的图像来说，处理的工作量是十分艰巨的，如 $M=N=512$ 时，$\boldsymbol{H}$ 矩阵的大小为 $MN\times MN=(512)^2\times(512)^2=262144\times262144$ ，求解 f 需要解 262144 个联立方程组，计算量之大难以想象。为解决这样的问题，须研究一些简单算法，利用$\boldsymbol{H}$ 矩阵的循环性质，使简化运算得以实现。

根据有关的数学知识，由于 $\boldsymbol{H}$ 是分块循环矩阵，则 $\boldsymbol{H}$ 可对角化，即

$$\boldsymbol{H}=\boldsymbol{WDW}^{-1} \tag{3-26}$$

$\boldsymbol{W}$ 为一变换矩阵，大小为 $MN\times MN$ 维，它由 $M\times M$ 个大小为 $N\times N$ 的子块的部分组成

$$\boldsymbol{W}=\begin{pmatrix} w(0,0) & w(0,1) & \cdots & w(0,M-1) \\ w(0,1) & w(1,1) & \cdots & w(1,M-1) \\ \vdots & \vdots & & \vdots \\ w(M-1,0) & w(M-1,1) & \cdots & w(M-1,M-1) \end{pmatrix} \tag{3-27}$$

其中

$$\boldsymbol{w}(i,m)=\exp\left[\mathrm{j}\frac{2\pi}{M}im\right]\boldsymbol{w}_N \tag{3-28}$$

式中，$i,m=0,1,2,\cdots,M-1$ ；$\boldsymbol{w}_N$为$N\times N$ 矩阵，其元素为

$$w_N(k,n)=\exp\left[\mathrm{j}\frac{2\pi}{N}kn\right] \tag{3-29}$$

式中，$k,n=0,1,2,\cdots,N-1$ 。

实际上，对任意形如 $\boldsymbol{H}$ 的分块循环矩阵，$\boldsymbol{W}$ 都可使其对角化。$\boldsymbol{D}$ 是对角阵，其对角元素与 $h_e(x,y)$ 的傅里叶变换有关，即如果

$$H(u,v)=\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}h_e(x,y)\exp\left(-\mathrm{j}2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right) \tag{3-30}$$

则 $\boldsymbol{D}$ 的 MN 个对角线元素按下面的形式给出，第一组 N 个元素为 $H(0,0),H(0,1),\cdots,H(0,N-1)$ ；第二组为 $H(0,0),H(1,1),\cdots,H(1,N-1)$ ；依此类推，最后的 N 个对角线元素为 $H(M-1,0),H(M-1,1),\cdots,H(M-1,N-1)$ 。由上述元素组成的整个矩阵再乘以 MN 得到 $\boldsymbol{D}$ ，即有

$$D(k,i)=\begin{cases} MNH\left(\left[\dfrac{k}{N}\right],k\bmod N\right) & i=k \\ 0 & i\neq k \end{cases} \tag{3-31}$$

式中，$\left[\dfrac{k}{N}\right]$ 表示不超过 $\dfrac{k}{N}$ 的最大整数；$k\bmod N$ 代表用 N 除 k 所得到的余数。

从而退化模型可写成

$$\boldsymbol{g}=\boldsymbol{Hf}+\boldsymbol{n}=\boldsymbol{WDW}^{-1}\boldsymbol{f}+\boldsymbol{n} \tag{3-32}$$

$$\boldsymbol{W}^{-1}\boldsymbol{g}=\boldsymbol{DW}^{-1}\boldsymbol{f}+\boldsymbol{W}^{-1}\boldsymbol{n} \tag{3-33}$$

可以证明

$$\boldsymbol{W}^{-1}\boldsymbol{g}=Vec[G(u,v)] \tag{3-34}$$

$$\boldsymbol{W}^{-1}\boldsymbol{f}=Vec[F(u,v)] \tag{3-35}$$

$$\boldsymbol{W}^{-1}\boldsymbol{n}=Vec[N(u,v)] \tag{3-36}$$

式中，$Vec[\cdot]$是将矩阵拉伸为向量的算子，例如

$$Vec\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}=\begin{pmatrix}1\\2\\3\\4\end{pmatrix}$$

$G(u,v)$、$F(u,v)$和$N(u,v)$分别是$g(x,y)$、$f(x,y)$和$n(x,y)$的二维傅里叶变换。于是有

$$G(u,v)=MNH(u,v)F(u,v)+N(u,v) \tag{3-37}$$

这样就将求$f(x,y)$的过程转换为求解$F(u,v)$的过程，简化了计算过程，同时上式也是进行图像复原的基础。

3.3 非约束复原

图像复原的主要目的是在假设具备退化图像 g 及 H 和 n 的某些知识的前提下，估计出原始图像 f 的估计值$\hat{f}$，估计值$\hat{f}$应使准则为最优（常用最小）。如果仅仅要求某种优化准则为最小，不考虑其他任何条件约束，这种复原方法为非约束复原法。

3.3.1 非约束复原的代数方法

由前面介绍的图像退化模型可知，其噪声项为

$$\boldsymbol{n}=\boldsymbol{g}-\boldsymbol{Hf} \tag{3-38}$$

在并不了解噪声项 $\boldsymbol{n}$ 的情况下，希望找到一个 $\boldsymbol{f}$，使得 $\boldsymbol{Hf}$ 在最小二乘方意义上来说近似于 $\boldsymbol{g}$，也就是说，希望找到一个$\boldsymbol{f}$的估计$\hat{\boldsymbol{f}}$，使得

$$\|\boldsymbol{n}\|^2=\|\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}\|^2 \tag{3-39}$$

为最小，由定义可知

$$\|\boldsymbol{n}\|^2=\boldsymbol{n}^{\mathrm{T}}\boldsymbol{n}$$
$$\|\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}\|^2=(\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}})^{\mathrm{T}}(\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}) \tag{3-40}$$

求$\|\boldsymbol{n}\|^2$最小等效于求$\|\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}\|^2$最小，即

$$J(\hat{\boldsymbol{f}})=\|\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}\|^2 \tag{3-41}$$

实际上是求$J(\hat{\boldsymbol{f}})$的极小值问题。求式（3-41）极小值的方法可以采用一般的求极值的方法进行处理。把$J(\hat{\boldsymbol{f}})$对$(\hat{\boldsymbol{f}})$微分，并使结果为零，即

$$\frac{\partial J(\hat{\boldsymbol{f}})}{\partial \hat{\boldsymbol{f}}}=-2\boldsymbol{H}^{T}(\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}})=0 \tag{3-42}$$

由式（3-42）可推导出

$$H^{\mathrm{T}}H\hat{f}=H^{\mathrm{T}}g \tag{3-43}$$

即

$$\hat{f}=(H^{\mathrm{T}}H)^{-1}H^{\mathrm{T}}g \tag{3-44}$$

令 $M=N$，因此，H 为一方阵，并且设 H^{-1} 存在，则可求得 $\hat{f}$，即

$$\hat{f}=H^{-1}(H^{\mathrm{T}})^{-1}H^{\mathrm{T}}g=H^{-1}g \tag{3-45}$$

以上就是在非约束条件下，利用最小二乘方准则和线性代数的方法，求取原图像信号的最近似值 $\hat{f}$ 的推导过程。

3.3.2 逆滤波复原法

逆滤波复原法也叫做反向滤波法，其主要过程是首先将要处理的数字图像从空间域转换到傅里叶频率域中，进行反向滤波后再由频率域转换到空间域，从而得到复原的图像信号，基本原理如下。

如果退化图像为 $g(x,y)$，原始图像为 $f(x,y)$，在不考虑噪声的情况下，其退化模型用式（3-11）表示，现在将其重写如下

$$g(x,y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(\alpha,\beta)h(x-\alpha,y-\beta)\mathrm{d}\alpha\mathrm{d}\beta$$

上式两边进行傅里叶变换得

$$G(u,v)=F(u,v)H(u,v) \tag{3-46}$$

式中，$G(u,v)$，$H(u,v)$，$F(u,v)$ 分别是退化图像 $g(x,y)$，点扩散函数 $h(x,y)$，原始图像 $f(x,y)$ 的傅里叶变换。

由式（3-46）以及傅里叶逆变换公式可得

$$F(u,v)=\frac{G(u,v)}{H(u,v)} \tag{3-47}$$

$$f(x,y)=F^{-1}[F(u,v)]=F^{-1}\left[\frac{G(u,v)}{H(u,v)}\right] \tag{3-48}$$

式中，$H(u,v)$ 可以理解为成像系统的“滤波”传递函数。在频域中系统的传递函数与原图像信号相乘实现“正向滤波”，这里 $G(u,v)$ 除以 $H(u,v)$ 起到了“反向滤波”的作用。这意味着，如果已知退化图像的傅里叶变换和“滤波”传递函数，则可以求得原始图像的傅里叶变换，经反傅里叶变换就可求得原始图像 $f(x,y)$。这就是逆滤波复原法的基本原理。

前面为了分析问题的简化，没有考虑噪声的影响。在有噪声的情况下，逆滤波复原法的基本原理可写成如下形式

$$G(u,v)=F(u,v)H(u,v)+N(u,v) \tag{3-49}$$

$$F(u,v)=\frac{G(u,v)-N(u,v)}{H(u,v)} \tag{3-50}$$

式中，$N(u,v)$ 是噪声 $n(x,y)$ 的傅里叶变换。

由于在逆滤波复原公式（3-48）中，$H(u,v)$ 处于分母的位置上，利用式（3-49）和式（3-50）进行图像复原处理时可能会发生下列情况：即在 u,v 平面上有些点或区域会产生 $H(u,v)=0$ 或 $H(u,v)$ 非常小的情况，在这种情况，即使没有噪声，也无法精确地恢复

$f(x,y)$。另外，在有噪声存在时，在 $H(u,v)$ 的领域内，$H(u,v)$ 的值可能比 $N(u,v)$ 的值小得多，因此由式（3-50）得到的噪声项可能会非常大，这样也会使 $f(x,y)$ 不能正确恢复。

一般来说，逆滤波复原法不能正确地估计 $H(u,v)$ 的零点，因此必须采用一个折中的方法进行解决。实际上，逆滤波不是用 $1/H(u,v)$，而是采用另外一个关于 u,v 的函数 $M(u,v)$。它的处理框图如图 3-5 所示。

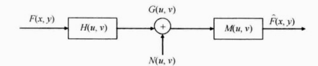


图 3-5　实际的逆滤波处理框图

在没有零点并且也不存在噪声的情况下，有

$$M(u,v)=\frac{1}{H(u,v)} \tag{3-51}$$

图 3-5 的模型包括了退化和恢复运算。退化和恢复总的传递函数可用 $H(u,v)\ M(u,v)$ 来表示。此时有

$$\hat{F}(u,v)=[H(u,v)M(u,v)]F(u,v) \tag{3-52}$$

式中，$\hat{F}(u,v)$ 是 $F(u,v)$ 的估计值；$\hat{F}(u,v)$ 是 $\hat{f}(u,v)$ 的傅里叶变换；$H(u,v)$ 是输入传递函数；$M(u,v)$ 是处理传递函数；$H(u,v)\ M(u,v)$ 是输出传递函数。

一般情况下，可以将图像的退化过程视为一个具有一定带宽的带通滤波器，随着频率的升高，该滤波器的带通特性很快下降，即 $H(u,v)$ 的幅度随着 u，v 平面原点的距离的增加而迅速下降，而噪声项 $N(u,v)$ 的幅度变化是比较平缓的。在远离 u，v 平面的原点时 $N(u,v)/H(u,v)$ 的值就会变得很大，而对于大多数图像来说，$F(u,v)$ 却变小，在这种情况下，噪声反而占优势，自然无法满意地恢复出原始图像。这一规律说明，应用逆滤波时仅在原点领域内采用 $1/H(u,v)$ 方能有效。换句话说，应使 $M(u,v)$ 在下述范围内选择

$$M(u,v)=\begin{cases}\dfrac{1}{H(u,v)} & u^2+v^2\leqslant w_0^2\\ 1 & u^2+v^2>w_0^2\end{cases} \tag{3-53}$$

式中，w_0 的选择应该将 $H(u,v)$ 的零点排除在此领域之外。

实验证明，当变质图像的信噪比较高，如信噪比 SNR=1000 或更高，而且轻度变质时，逆滤波复原方法可以获得较好的效果。

3.4　有约束复原

为了在数学上更容易处理，通常在无约束复原方法的基础上附加一定的约束条件，从而在多个可能结果中选择一个最佳结果，这便是有约束复原方法。

3.4.1 最小二乘类约束复原

无约束复原是除了使准则函数 $J(\hat{f})=\|g-H\hat{f}\|^2$ 最小外，再没有其他约束条件。因此只需要了解退化系统的传递函数或点扩展函数 H，就能利用式（3-44）或式（3-45）进行复原。但是由于传递函数 H 奇异性的问题，复原只能局限在靠近原点的有限区域内进行，这就使得无约束复原方法具有较大的局限性。

最小二乘类约束复原是指除了要求了解关于退化系统的传递函数 H 之外，还需要知道某些噪声的统计特性或噪声与图像的某些相关情况。根据所了解的噪声先验知识的不同，应采用不同的约束条件，可得到不同的图像复原技术。

在最小二乘类约束复原中，复原问题表现为在满足 $\|\boldsymbol{n}\|^2=\|\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}\|^2$ 的约束条件下，要设法寻找一个最优估计 $\hat{\boldsymbol{f}}$，使得形式为 $\|\boldsymbol{Q}\hat{\boldsymbol{f}}\|^2=\|\boldsymbol{n}\|^2$ 的函数最小化。对于这类问题的有约束最小化问题，通常采用拉格朗日乘数法进行处理。即寻找一个 $\hat{f}$，使得如下准则函数最小。

$$J(\hat{\boldsymbol{f}})=\|\boldsymbol{Q}\hat{\boldsymbol{f}}\|^2+\lambda\left(\|\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}\|^2-\|\boldsymbol{n}\|^2\right) \tag{3-54}$$

式中，$\boldsymbol{Q}$ 为 $\hat{\boldsymbol{f}}$ 的线性算子，λ 为一常数（称为拉格朗日乘子）。对式（3-54）求导，可得

$$\frac{\partial}{\partial\hat{\boldsymbol{f}}}J(\hat{\boldsymbol{f}})=2\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\hat{\boldsymbol{f}}-2\lambda\boldsymbol{H}^{\mathrm{T}}(\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}})=0 \tag{3-55}$$

$$\hat{\boldsymbol{f}}=\left(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}+\frac{1}{\lambda}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\right)^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{g} \tag{3-56}$$

令 $\gamma=1/\lambda$，得

$$\hat{\boldsymbol{f}}=(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}+\gamma\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{g} \tag{3-57}$$

常数 λ 必须反复迭代齐整，直到满足约束条件 $\|\boldsymbol{n}\|^2=\|\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}\|^2$。求解式（3-57）的关键就是如何选用一个合适的变换矩阵 $\boldsymbol{Q}$。

相对于无约束问题，有约束条件的图像复原更符合图像退化的实际情况，因此其适应面更加广泛。对式（3-57），若选择不同形式的 $\boldsymbol{Q}$ 矩阵，则可得到不同类型的有约束最小二乘方类图像复原方法。如果采用图像 f 和噪声的自相关矩阵 $\boldsymbol{R}_f$ 和 $\boldsymbol{R}_n$ 表示 $\boldsymbol{Q}$，就可以得到维纳滤波复原方法。若采用拉普拉斯算子形式，即使某个函数的二阶导数最小，那么也可推导出有约束最小平方复原方法。

下面是最小二乘滤波复原的 MATLAB 应用，如图 3-6 所示。

程序代码如下：

```
I=imread('F:\image\lena.bmp'); %读取原始图像
LEN=31;
%图像的模糊化
THETA=11;
PSF1=fspecial('motion',LEN,THETA);
PSF2=fspecial('gaussian',10,5);
```

```
Blurred1=imfilter(I,PSF1,'circular','conv');
Blurred2=imfilter(I,PSF2,'conv');
%模糊化图像加噪
V=.002;
BlurredNoisy1=imnoise(Blurred1,'gaussian',0,V);
BlurredNoisy2=imnoise(Blurred2,'gaussian',0,V);
figure,
subplot(1,3,1);imshow(I)
title(' lena ')
%用真实的 PSF 函数和噪声强度作为参数进行图像复原
NP=V*prod(size(I));
reg1=deconvreg(BlurredNoisy1,PSF1,NP);
reg2=deconvreg(BlurredNoisy2,PSF2,NP);
figure;
subplot(1,3,2);imshow(reg1);
title('Restored1 with NP')
figure;
subplot(1,3,3) ;imshow(reg2);
title('Restored2 with NP')
```


a)


b)


c)

图 3-6　用真实的 PSF()函数和噪声强度作为参数的约束最小二乘滤波复原

a) 原始图像　b) BlurredNoisy1—"motion"　c) BlurredNoisy2—"gaussian"

3.4.2 维纳滤波

在一般情况下，图像信号可近似为平稳随机过程，维纳滤波的基本原理是将原始图像 f 和对原始图像的估计 $\hat{f}$ 看为随机变量，按照使 f 和估计值 $\hat{f}$ 之间的均方误差达到最小的准则实现图像复原，即

$$e^2 = E\{[f(x,y) - \hat{f}(x,y)]^2\} \tag{3-58}$$

式中，$E[\cdot]$ 表示数学期望。

设 $\boldsymbol{R}_f$ 和 $\boldsymbol{R}_n$ 分别是 $\boldsymbol{f}$ 和 $\boldsymbol{n}$ 的自相关矩阵，定义如下：

$$\boldsymbol{R}_f = E(\boldsymbol{f}\boldsymbol{f}^{\mathrm{T}}) \tag{3-59}$$

$$\boldsymbol{R}_n = E(\boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}) \tag{3-60}$$

根据上述定义可知，$\boldsymbol{R}_f$ 和 $\boldsymbol{R}_n$ 均为实对称矩阵。在大多数实际图像中，相近像素点是高度相关的，而距离较远的像素点的相关性则相对较弱。通常情况下，无论是 $\boldsymbol{f}$ 还是 $\boldsymbol{n}$，其元素之间的相关不会延伸到 20～30 个像素的距离之外。因此，一般来说，自相关矩阵 $\boldsymbol{R}_f$ 和 $\boldsymbol{R}_n$ 在主对角线附近有一个非零元素区域，而矩阵的右上角和左上角的区域内将接近零值。如果像素之间的相关是像素距离的函数，而不是像素位置的函数，则可将 $\boldsymbol{R}_f$ 和 $\boldsymbol{R}_n$ 近似为分块循环矩阵。因而，用循环矩阵的对角化，可写成如下形式：

$$\boldsymbol{R}_f = \boldsymbol{WAW}^{-1} \tag{3-61}$$

$$\boldsymbol{R}_n = \boldsymbol{WBW}^{-1} \tag{3-62}$$

$\boldsymbol{W}$ 为 $MN \times MN$ 矩阵，包含 $M \times M$ 个 $N \times N$ 子矩阵。

以 $\boldsymbol{W}(i,m)$ 表示 $\boldsymbol{W}$ 的 i 和 m 列分块矩阵，则

$$\boldsymbol{W}(i,m) = \mathrm{e}^{\mathrm{j}\frac{2\pi}{M}im}\boldsymbol{W}_N \qquad i,m = 0,1,2,\cdots,M-1 \tag{3-63}$$

$\boldsymbol{W}_N$ 是 $N \times N$ 矩阵，以 $W(k,n)$ 表示 k 行 n 列元素，则有

$$W_N(k,n) = \mathrm{e}^{\mathrm{j}\frac{2\pi}{M}kn} \qquad k,n = 0,1,2,\cdots,M-1 \tag{3-64}$$

矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$ 的元素分别为矩阵 $\boldsymbol{R}_f$ 和 $\boldsymbol{R}_n$ 中的自相关元素的傅里叶变换，这些自相关的傅里叶变换分别定义为 $f_e(x,y)$ 和 $n_e(x,y)$ 的谱密度 $R_f(u,v)$ 和 $P_n(u,v)$。

定义 $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \boldsymbol{R}_f^{-1}\boldsymbol{R}_n$ 代入式（3-57），则

$$\begin{aligned}\hat{\boldsymbol{f}} &= (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H} + \gamma\boldsymbol{R}_f^{-1}\boldsymbol{R}_n)^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{g} \\ &= (\boldsymbol{WD}^*\boldsymbol{DW}^{-1} + \gamma\boldsymbol{WA}^{-1}\boldsymbol{BW}^{-1})^{-1}\boldsymbol{WD}^*\boldsymbol{W}^{-1}\boldsymbol{g}\end{aligned}$$

因此可得

$$\boldsymbol{W}^{-1}\hat{\boldsymbol{f}} = (\boldsymbol{D}^*\boldsymbol{D} + \gamma\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{D}^*\boldsymbol{W}^{-1}\boldsymbol{g} \tag{3-65}$$

若 $M = N$，则有

$$\begin{aligned}\hat{F}(u,v) &= \frac{|H(u,v)|}{|H(u,v)|^2 + \gamma\dfrac{P_n(u,v)}{P_f(u,v)}}G(u,v) \\ &= \frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + \gamma\dfrac{P_n(u,v)}{P_f(u,v)}}G(u,v) \qquad u,v = 0,1,2,\cdots,N-1\end{aligned} \tag{3-66}$$

若 $\gamma = 1$，则称为维纳滤波器，当无噪声影响时，由于 $P_n(u,v) = 0$，则退化为逆滤波器，又称为理想的逆滤波器，因此，逆滤波器是维纳滤波器的一种特殊情况。需要注意的是，$\gamma = 1$ 并不是在有约束条件下的最佳解，此时并不满足约束条件 $\|\boldsymbol{n}\|^2 = \|\boldsymbol{g} - \boldsymbol{H}\hat{\boldsymbol{f}}\|^2$。若 γ 为变参数，则称为变参数维纳滤波器。Slepian 将维纳去卷积推广用于处理卷积计算效率的方法。

维纳去卷积提供了一种在有噪声情况下导出去卷积传递函数的最优方法，但下面的三个问题限制了它的有效性。

1）当图像复原的目的是供人观察时，方均误差（MSE）准则并不是一个特别好的优化准则。这是因为 MSE 准则不管其在图像中的位置，对所有误差都赋予同样的权，而人眼则

对暗处和高梯度区域的误差比其他区域的误差具有较大的容忍性。由于使方均误差最小化，因此维纳滤波器以一种并非最适合人眼的方式对图像进行了平滑。

2）经典的维纳去卷积不能处理具有空间可变点扩散函数的情形，如存在彗差、散差、表面像场弯曲以及包含旋转的运动模糊等情况。

3）这种技术不能处理非平稳信号和噪声的一般情形。许多图像都是高度非平衡的，有着被陡峭边缘分开的大块平坦区域。此外，一些重要的噪声源具有与局部灰度有关的特性。

下面是维纳滤波复原的 MATLAB 应用，如图 3-7 所示。


a)


b)


c)

图 3-7 采用真实 PSF()函数的维纳滤波复原（无噪声）

a) 原始图像 b) Blurred—“motion” c) Blurred—“gaussian”

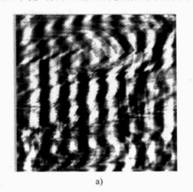
程序代码如下：

```
I=imread('lena.bmp');
LEN=31;
THETA=11;
PSF1=fspecial('motion',LEN,THETA);
PSF2=fspecial('gaussian',10,5);
Blurred1=imfilter(I,PSF1,'circular','conv');
Blurred2=imfilter(I,PSF2,'conv');
V=.002;
BlurredNoisy1=imnoise(Blurred1,'gaussian',0,V);
BlurredNoisy2=imnoise(Blurred2,'gaussian',0,V);
figure,
subplot(1,3,1);imshow(I);
title('lena');
wnr1=deconvwnr(Blurred1,PSF1);
wnr2=deconvwnr(Blurred1,PSF2);
figure;
subplot(1,3,2);imshow(wnr1);
title('Restored1,True PSF');
figure;
subplot(1,3,3) ;imshow(wnr2);
```



```
title('Restored2,True PSF')
```

从复原的图像来看，对于运动引起的模糊图像的复原效果要好于因 gaussian 模糊的图像复原的效果。但这是假定预先知道引起模糊的精确点扩散函数，实际图像处理时，大多数情况下无法准确得知精确点扩散函数，一般采用估计的 PSF()函数来复原图像。下面的例子是针对运动引起的图像模糊，利用估计的点扩散函数，相对于真实的点扩散函数，分别采用了过大的模糊距离参数和过大的模糊运动方向角度参数，如图 3-8 所示。


a)


b)

图 3-8 利用估计的 PSF()函数复原模糊图像（无噪声）

a) 过大的模糊距离参数 b) 过大的模糊运动方向角度参数

```
BlurredNoisy1=imnoise(Blurred1,'gaussian',0,V);
BlurredNoisy2=imnoise(Blurred2,'gaussian',0,V);
wnr3=deconvwnr(Blurred1,PSF1);
wnr3=deconvwnr(Blurred1,fspecial('motion',2*LEN,THETA));
wnr4=deconvwnr(Blurred1,fspecial('motion',LEN,2*THETA));
figure;
subplot(1,2,1);imshow(wnr3);
title('Restored1,True PSF');
figure;
subplot(1,2,2);imshow(wnr2);
title('Restored2,True PSF')
```

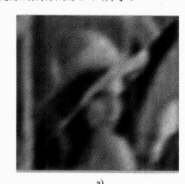
除了利用以上参数进行图像复原，MATLAB 还提供了利用图像的自相关信息来提高图像复原质量的方法，这需要提供信号的自相关函数 ICORR()和噪声的自相关函数 NCORR()。

程序代码如下：

```
NP=(V*prod(size(I))).^2;
NPOW=sum(NP(:))/prod(size(I)); %噪声功率
NCORR=fftshift(real(ifftn(NP))); %噪声自相关函数（ACF）
IP=abs(fftn(im2double(I))).^2;
IPOW=sum(IP(:))/prod(size(I));%原始图像的功率
ICORR=fftshift(real(ifftn(IP))); %图像自相关函数（ACF）
wnr5=deconvwnr(BlurredNoisy1,PSF1,NCORR,ICORR);
```

```
wnr6=deconvwnr(BlurredNoisy2,PSF2,NCORR,ICORR);
figure;
subplot(1,2,1);imshow(wnr5);
subplot(1,2,2);imshow(wnr6)
```

图像复原结果如图 3-9 所示。



a)　　　　b)

图 3-9　利用自相关信息进行图像复原

a) BlurredNoisy—"motion"　b) BlurredNoisy—"gaussian"

3.4.3 Lucy_Richardson 滤波复原

Lucy_Richardson 算法是目前应用最广泛的图像复原技术之一，采用迭代的方法。Lucy_Richardson 算法能够按照泊松噪声统计标准求出与给定 PSF 卷积后，最有可能成为输入模糊图像的图像。当 PSF 已知，但图像噪声信息未知时，也可以使用这个函数进行有效的工作。从成像方程和 Poissian 统计可以有如下推导：

$$I(i)=\sum_i P(i\setminus j)O(j) \tag{3-67}$$

式中，O 是原始图像；$P(i\setminus j)$ 是 PSF()函数；I 是无噪声模糊图像。在已知 $I(i)$ 时，在每个像素点估计 $D(i)$ 的联合似然函数为

$$\ln\prod=\sum_i D(i)\ln I(i)-I(i)-\ln D(i)! \tag{3-68}$$

当式（3-68）存在时，最大联合似然函数的解存在。

$$\frac{\partial \ln\prod}{\partial O(j)}=\sum_i\left[\frac{D(i)}{I(i)}-1\right]P(i\setminus j)=0 \tag{3-69}$$

则可得 Lucy_Richardson 迭代式，即

$$O_{\text{new}}(j)=O(j)\sum_i P(i\setminus j)\frac{D(i)}{I(i)}\Big/\sum_i P(i\setminus j) \tag{3-70}$$

可以看出每次迭代时，都可以提高解的似然性，随着迭代次数的增加，最终将会收敛在具有最大似然性的解处。

MATLAB 提供的 deconvlucy()函数，就是利用加速收敛的 Lucy_Richardson 算法对图像进行复原。deconvlucy()函数还能够用于实现复杂图像重建的多种算法中，这些重建算法都是

基于原始 Lucy_Richardson 最大化可能性算法。deconvlucy()函数的调用方式如下：

- J=deconvlucy(I,PSF)
- J=deconvlucy(I,PSF,NUMIT)
- J=deconvlucy(I,PSF,NUMIT,DAMPAR)
- J=deconvlucy(I,PSF,NUMIT,DAMPAR,WEIGHT)
- J=deconvlucy(I,PSF,NUMIT,DAMPAR,WEIGHT,READOUT)
- J=deconvlucy(I,PSF, NUMIT,DAMPAR,WEIGHT,READOUT,SUBSMPL)



其中，I 表示输入图像。PSF 表示点扩散函数。其他参数都是可选参数：NUMIT 表示算法的重复次数，默认值为 10；DAMPAR 表示偏差阈值，默认值为 0（无偏差）；WEIGHT 表示像素加权值，默认值为原始图像的数值；READOUT 表示噪声矩阵，默认值为 0；SUBSMPL 表示子采样时间，默认值为 1。

deconvlucy()函数的输出 J 是一个单元数组，包含 4 个元素：output(1)，原始输入图像；output(2)，最后一次反复产生的图像；output(3)，倒数第二次产生的图像，output(4)，deconvlucy 函数用来获知重新起始点的内部信息。输出的单元数组可以作为输入参数传递给 deconvlucy 函数，从而重新开始反复计算过程。

噪声痕迹是最大化可能性数据逼近算法的常见问题。经过多次重复处理，尤其是在低信噪比条件下，重建图像可能会出现一些斑点，这些斑点并不代表图像的真实结构，是输出图像过于逼近噪声所产生的结果。要有效控制这些痕迹，可以使用 deconvlucy 函数的收敛参数 DAMPAR，该参数指定了收敛过程中结果图像与原始图像背离程度的阈值。对于那些超过阈值的数据，将不再允许进行反复计算。

图像复原的另一复杂之处是那些可能包括坏像素的数据，可能会随时间和位置的变化而变化。这样，可以通过 deconvlucy()函数的 WEIGHT 数组参数，指定图像中可以忽略的某些像素，将需要忽略的像素对应 WEIGHT 数组元素，设置为 0。

检测器中的噪声由两部分组成，一是呈泊松分布的光子计算噪声；一是镜头呈高斯分布的读取噪声。在利用 Lucy_Richardson 算法进行图像复原的过程中可以声明第一种类型的噪声，但是第二种噪声情况用户必须自己声明，deconvlucy()函数使用 READOUT 参数指定噪声类型，其值通常是读取噪声变量和背景噪声变量的总和，其数值的大小将指定能够确保所有数值为正数的偏移量。

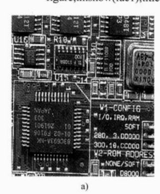
如果对采样不足的数据进行重建，而重建过程建立在一个较好的网格操作基础上，则重建效果就可以大大提高。如已知 PSF 具有较高的分辨率，则 deconvlucy()函数使用 SUBSMPL 参数指定采样不足的比例。

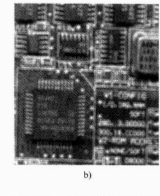
另外，PSF 还可以通过观察像素偏移或光学模型技术获得。这种方法对高信噪比图像尤为有效，因为目标图像可以被有效地限制在像素的中心位置。如果图标图像位于两个像素之间，那么它将被作为领域像素的组合进行重建。一个好的网络将会使目标图像扩散流序列重新朝向图像的中心。

下面给出实例说明 deconvlucy()函数的具体使用方法。

以下程序段首先读取原始电路板图像，如图 3-10a 所示，然后对原始图像进行加载模糊化，仿真传输失真现象，如图 3-10b 所示。对模糊图像调用 deconvlucy()函数进行图像复原，得到如图 3-10c 所示的结果。

```
I=imread('board.tif');
I=I(50+[1:256],2+[1:256],:);
figure;imshow(I);title('Original Image');
PSF=fspecial('gaussian',5,5);
Blurred=imfilter(I,PSF,'symmetric','conv');
V=.002;
BlurredNoisy=imnoise(Blurred,'gaussian',0,V);
figure;imshow(BlurredNoisy);title('Blurred & Noisy');
luc1=deconvlucy(BlurredNoisy,PSF,5);
figure;imshow(luc1);title('Restored Image,NUMIT=5');
```


a)


b)

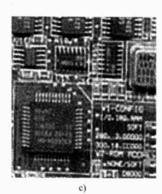

c)

图 3-10　原始图像加噪及恢复

a) 原始图像　b) 模糊加噪　c) 恢复图像

deconvlucy()函数的其他参数调用方法及其效果与前面两个图像复原函数类似，不再赘述。

3.4.4 盲解卷积复原

前面几种图像复原方法都是在知道模糊图像的点扩展函数的情况下进行的，而在实际应用中，通常都要在不知道点扩展函数的情况下进行图像复原。盲解卷积复原就是在这种应用背景下提出的。盲解卷积复原是利用原始模糊图像，同时估计 PSF 和清晰图像的一种图像复原方法。具体实现算法有先验模糊辨识方法、非参数限定支持域恢复方法，以及 ARMA 参数估计方法、基于高阶统计量的非参数法等。

MATLAB 7.0 提供了 deconvblind()函数用于实现盲解卷积，该函数类似于加速收敛 Lucy-Richardson 算法的执行过程，同时要重建图像和 PSF。盲解卷积算法一个很好的优点就是，在对失真情况（包括噪声和模糊）毫无先验知识的情况下，仍然能够实现对模糊图像的复原操作。同时，deconvblind()函数与 deconvlucy()函数一样，也可以用于实现多种复杂图像重建修改算法，而这些算法都是以原始 Lucy-Richardson 最大化可能性算法为基础的。

deconvblind 函数的调用格式如下：

- [J, PSF]= deconvblind(I, INITPSF)
- [J, PSF]= deconvblind(I, INITPSF, NUMIT)

- [J, PSF]= deconvblind(I, INITPSF, NUMIT,DAMPAR)
- [J, PSF]= deconvblind(I, INITPSF, NUMIT, DAMPAR, WEIGHT)
- [J, PSF]= deconvblind(I, INITPSF, NUMIT, DAMPAR, WEIGHT, READOUT)

其中，I 表示输入图像，INITPSF 表示 PSF 的估计值，NUMIT 表示算法重复次数，DAMPAR 只表示偏移阈值，WEIGHT 用来屏蔽坏像素，READOUT 表示噪声矩阵。输出参数 J 表示复原后的图像，PSF 与 INITPSF 具有相同的大小，表示重建点扩散函数。

下面将通过几个程序代码实例来说明 deconvblind()函数的使用方法。

1. 图像模糊化

首先利用以下程序段，读取原始图像，如图 3-11 所示，然后生成如图 3-12 所示的点扩展函数 PSF，再对图像进行模糊化，模糊结果如图 3-13 所示。



图 3-11　原始图像

图 3-12　真实 PSF



图 3-13　模糊后的图像

```
I=imread('cameraman.tif');
figure;imshow(I);title('Original Image');
PSF=fspecial('motion',13,45);
figure;imshow(PSF,[]);title('True PSF');
Blurred=imfilter(I,PSF,'circ','conv');
figure;imshow(Blurred);title('Blurred Image');
```

2. 图像复原

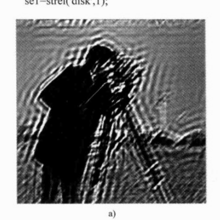
在调用 deconvblind()函数进行图像复原时，INITPSF 的大小是非常重要的一个指标。在实际应用中，通过分析，都是使用不同大小的 PSF 对图像进行重建，从中选择一个最合适的 PSF 值。以下程序段以真实大小的 INITPSF 进行图像复原，得到初步复原结果，如图 3-14a 所示，同时初步重建 PSF，如图 3-14b 所示。

```
INITPSF=ones(size(PSF));
[J P]=deconvblind(Blurred,INITPSF,30);
figure;imshow(J);
figure;imshow(P,[],'notruesize');
```

下面是利用 WEIGHT 数组对权值矩阵图像进行重建，得到如图 3-15 所示的复原结果。由图可以看出，复原后的图像消除了“环”的存在，但是复原结果仍然有一定的失真。

程序代码如下：

```
WEIGHT=edge(I,'sobel',.28); %sobel 算子边缘提取
se1=strel('disk',1);
```


a)

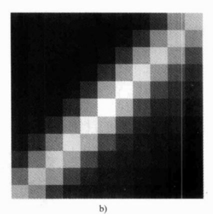
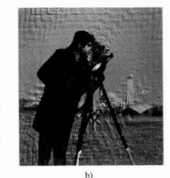

b)

图 3-14　初步复原的图像与初步重建使用的 PSF

a) 初步复原的图像　b) 初步重建使用的 PSF

```
se2=strel('line',13,45);
WEIGHT=1-double(imdilate(WEIGHT,[se1 se2]));  %膨胀操作，边界像素设为零
figure;imshow(WEIGHT);
P1=P; %保存数据
P1(find(P1<0.01))=0; %修改 PSF()函数
[J2 P2]=deconvblind(Blurred,P1,50,[],WEIGHT) %利用上面得到的 WEIGHT 进行盲解卷积
figure;imshow(J2);
figure;imshow(P2,[],'notruesize');
```


a)


b)

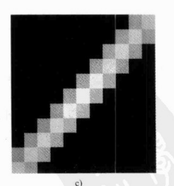

c)

图 3-15　图像复原及使用的 PSF

a) 权值矩阵　b) 图像复原　c) 图像复原使用的 PSF

3.5 几种其他图像复原技术

前边已经讨论了几种基本的代数图像复原技术。除此之外，尚存在一些其他的空间图像复原方法，本节将对这些方法进行一些简单的讨论。

3.5.1 几何畸变校正

在图像的获取或显示过程中往往会产生几何畸变。例如，成像系统有一定的几何非线性。这主要是由于摄像管、摄像机及阴极射线管显示器的扫描偏转系统有一定的非线性，因此会形成如图 3-16b、c 所示的畸变图像。图 3-16a 为原始图像，图 3-16b 和 3-16c 分别为枕形畸变和桶形畸变。

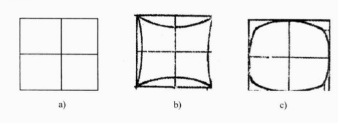


图 3-16 几何畸变

a) 原始图像 b) 枕形畸变 c) 桶形畸变

除此之外，还有由于斜视角度获得的图像的透视畸变。另外，由卫星摄取的地球表面的图像往往覆盖较大的面积，由于地球表面呈球形，这样摄取的平面图像也将会有较大的几何畸变。对于这些图像必须加以校正，以免影响分析精度。

由成像系统引起的几何畸变的校正有两种方法。一种是预畸变法，这种方法是采用与畸变相反的非线性扫描偏转法，用来抵消预计的图像畸变；另一种是所谓的后验校正方法，这种方法是用多项式曲线在水平和垂直方向去拟合每一条畸变的网线，然后求得反变化的校正函数。用这个校正函数即可校正几何畸变的图像。图像的空间几何畸变及其校正过程如图 3-17 所示。

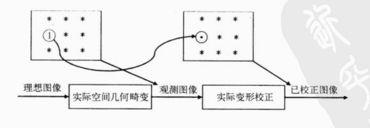


图 3-17 空间几何畸变及其校正的过程

任意几何畸变都可以由非畸变坐标系 (x, y) 变换到畸变坐标系 (x', y') 的方程来定义。方程的一般形式为

$$\begin{cases} x' = h_1(x,y) \\ y' = h_2(x,y) \end{cases} \tag{3-71}$$

在透视畸变的情况下，变换是线性的，即

$$\begin{cases} x' = ax + by + c \\ y' = dx + ey + f \end{cases}$$

设 $f(x,y)$ 是无畸变的原始图像，而 $g(x',y')$ 是 $f(x,y)$ 畸变结果，这一畸变的过程是已知的，并且用函数 h_1 和 h_2 定义。于是有

$$g(x',y') = f(x,y) \tag{3-72}$$

这说明在图像中本来应该出现在像素 (x,y) 上的灰度值由于畸变，实际上却出现在 (x',y') 上了。这种失真的复原问题实际上是映射变换问题。在给定了 $g(x',y')$，$h_1(x,y)$，$h_2(x,y)$ 的情况下，其复原处理可按如下的方法进行。

1）对于 $f(x,y)$ 中的每一点 (x_0,y_0)，找出在 $g(x',y')$ 中相应的位置 $(\alpha,\beta) = [h_1(x_0,y_0)\ h_2(x_0,y_0)]$。由于 α 和 β 不一定是整数，所以通常 (α,β) 不会与 $g(x',y')$ 中的任何点重合。

2）找出 $g(x',y')$ 中与 (α,β) 最靠近的点 (x',y')，并且令 $f(x_0,y_0) = g(x_1',y_1')$，也就是把 $g(x',y')$ 点的灰度值赋予 $f(x_0,y_0)$。如此逐点做下去，直到整个图像，则几何畸变得到校正。

3）如果不采用 2）中的灰度值的代换方法，也可以采用内插法。这种方法是假定 (α,β) 点找到后，在 $g(x',y')$ 中找出包围着 (α,β) 点的 4 个邻近的数字点，(x_1',y_1')，(x_{1+1}',y_1')，(x_1',y_{1+1}')，(x_{1+1}',y_{1+1}')，并且有

$$\begin{cases} x_1' \leqslant \alpha < x_{1+1}' \\ y_1' \leqslant \beta < y_{1+1}' \end{cases} \tag{3-73}$$

式中，$f(x,y)$ 中 (x_0,y_0) 点的灰度值由 $g(x',y')$ 中 4 个点的灰度值间的某种内插法来确定。

在以上方法的几何畸变校正处理中，如果 (α,β) 处在图像 $g(x',y')$ 之外，则不能确定其灰度值，而且校正后的图像多半不能保持其原来的矩形形状。

以上讨论的是 g，h_1，h_2 都知道的情况下几何畸变的校正方法。如果只知道 g，而 h_1 和 h_2 都不知道，但是若有类似的、规则的网格之类的图案可供参考利用，那么就有可能通过测量 g 中的网格点的位置来决定畸变变换的近似值。

例如，如果给出了 3 个邻近网格点构成的小三角形，其在规则网格中的理想坐标为 (r_1,s_1)，(r_2,s_2)，(r_3,s_3)，并设这些点在 g 中的位置分别为 (u_1,v_1)，(u_2,v_2)，(u_3,v_3)。由线性变换关系

$$\begin{cases} x' = ax + by + c \\ y' = dx + ey + f \end{cases} \tag{3-74}$$

可认为它把 3 个点映射到它们畸变后的位置，由此，可构成如下 6 个方程：

$$\begin{cases} u_1 = ar_1 + bs_1 + c \\ v_1 = dr_1 + es_1 + f \\ u_2 = ar_2 + bs_2 + c \\ v_2 = dr_2 + es_2 + f \\ u_3 = ar_3 + bs_3 + c \\ v_3 = dr_3 + es_3 + f \end{cases} \tag{3-75}$$

解这 6 个方程可以求得a,b,c,d,e,f。这种变换可用来校正 g 中被这 3 点连线包围的三角形畸变分部。由此对每 3 个一组的网格点重复进行，即可实现全部图像的几何畸变校正。

3.5.2 盲目图像复原



多数的图像复原技术都是以图像退化的某种先验知识为基础，也就是假定系统的脉冲响应或点扩散函数是已知的。但是，在许多情况下难以确定图像退化的点扩散函数。在这种情况下，必须从观察图像中以某种方式抽出图像退化信息，从而找出图像复原方法。这种方法就是所谓的盲目图像复原。对具有加性噪声的模糊图像作盲目图像复原的方法有两种，就是直接测量法和间接估计法。

直接测量法盲目图像复原通常要测量图像的模糊脉冲响应和噪声功率谱或协方差函数。在所观察的景物中，往往点光源能直接指示出冲激响应。另外，图像边缘是否陡峭也能用来推测模糊冲激响应。在背景高度相对恒定的区域内测量图像的协方差可以估计出观测图像的噪声协方差函数。

间接估计法盲目图像复原类似于多图像平均法处理。例如，在电视系统中，观测到的第 i 帧图像为

$$g_i(x,y)=f_i(x,y)+n_i(x,y) \tag{3-76}$$

式中，$f_i(x,y)$ 是原始图像；$g_i(x,y)$ 是含有噪声的图像；$n_i(x,y)$ 是加性噪声。如果原始图像在 M 帧观测图像内保持恒定，对 M 帧观测图像求和，得到的关系如下

$$f_i(x,y)=\frac{1}{M}\sum_{i=1}^{M}g_i(x,y)-\frac{1}{M}\sum_{i=1}^{M}n_i(x,y) \tag{3-77}$$

当 M 很大时，式（3-77）右边的噪声项的值趋向于它的数学期望值 $E\{n(x,y)\}$。一般情况下，白色高斯噪声在所有 (x,y) 上的数学期望都等于零，因此，合理的估计量是

$$\hat{f}_i(x,y)=\frac{1}{M}\sum_{i=1}^{M}g_i(x,y) \tag{3-78}$$

以上是利用多幅相同的图像进行平均以实现对加性噪声的消除，同时，盲目图像复原的间接估计法也可以利用时间上平均的概念去掉图像中的模糊。

3.6 运动模糊图像的复原

3.6.1 模糊模型

由于摄像机和景物之间的相对运动，往往造成获取的图像模糊。而且很多变速的非直线运动在一定条件下可以看成是由均匀直线运动合成的，因此由均匀直线运动所造成的模糊图像的复原问题更具有普遍意义。

设图像 $f(x,y)$ 作平面匀速直线运动，令 $x_0(t)$ 和 $y_0(t)$ 分别为 x 和 y 方向运动的随时间变化的分量。如照相机拍照，胶片上任一点上总曝光量是由快门打开的时间 T 内所有曝光的积分而得，设快门的开、关是瞬时发生的，且光学成像过程是完善的，则有

$$g(x,y)=\int_0^T f[x-x_0(t),y-y_0(t)]\mathrm{d}t \tag{3-79}$$

对式（3-79）进行傅里叶变换

$$G(u,v)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}g(x,y)\exp[-\mathrm{j}2\pi(ux,vy)]\mathrm{d}x\mathrm{d}y$$
$$=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{0}^{T}f[x-x_0(t),y-y_0(t)]\mathrm{d}t\exp[-\mathrm{j}2\pi(ux,vy)]\mathrm{d}x\mathrm{d}y \tag{3-80}$$

此处积分次序交换后得

$$G(u,v)=\int_0^T\left[\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f[x-x_0(t),y-y_0(t)]\exp[-\mathrm{j}2\pi(ux,vy)]\mathrm{d}x\mathrm{d}y\right]\mathrm{d}t$$
$$=\int_0^T\{F(u,v)\exp[-\mathrm{j}2\pi(ux_0(t)+vy_0(t))]\}\mathrm{d}t$$
$$=F(u,v)\int_0^T\exp\{-\mathrm{j}2\pi[ux_0(t)+vy_0(t)]\}\mathrm{d}t$$

令

$$H(u,v)=\int_0^T\exp\{-\mathrm{j}2\pi[ux_0(t)+vy_0(t)]\}\mathrm{d}t \tag{3-81}$$

则可得到

$$G(u,v)=H(u,v)F(u,v) \tag{3-82}$$

$$F(u,v)=\frac{G(u,v)}{H(u,v)} \tag{3-83}$$

因此，$f(x,y)$ 可由 $F(u,v)$ 的傅里叶变换求得。

3.6.2 水平匀速直线运动引起模糊的复原

如果模糊图像是由景物在 x 方向上进行水平匀速直线运动造成的，则模糊后图像上任意点的值为

$$g(x,y)=\int_0^T f[x-x_0(t),y]\mathrm{d}t \tag{3-84}$$

设图像总的位移为 a，总的运动时间为 T，则运动性质为 $x_0(t)=\frac{a}{T}t$，于是有

$$H(u,v)=\int_0^T\exp[-\mathrm{j}2\pi ux_0(t)]\mathrm{d}t=\int_0^T\exp[-\mathrm{j}2\pi u\frac{a}{T}t]\mathrm{d}t=\frac{T}{\pi ua}\sin(\pi ua)\exp(-\mathrm{j}\pi ua) \tag{3-85}$$

由式（3-85）可见，当 $u=\frac{n}{a}$（n 为整数）时，$H(u,v)=0$，在这些点上无法用逆滤波法恢复原图像，因而需采用其他方法。

由于只考虑 x 方向，y 是不变的，故可暂时忽略 y，式（3-84）可写成

$$g(x)=\int_0^T f[x-x_0(t)]\mathrm{d}t=\int_0^T f\left(x-\frac{a}{T}t\right)\mathrm{d}t \quad 0\leqslant x\leqslant L \quad (L\text{ 为图像的宽度}) \tag{3-86}$$

令 $\tau=x-\frac{a}{T}t$，则有

$$g(x)=\frac{T}{a}\int_{x-a}^{x}f(\tau)\mathrm{d}\tau$$

对上式两边求导，有

$$g'(x)=\frac{T}{a}[f(x)-f(x-a)]$$

$$f(x)=\frac{a}{T}g'(x)+f(x-a) \tag{3-87}$$

式（3-87）反映了 $f(x)$ 和 $f(x-a)$ 的递推关系。因为 $g'(x)$ 、T、a 是已知的，因此知道了长度为 a 的区间上的原始图像，就可以推出整幅图像，可想办法找出一种递归方法来复原图像。



由于图像在 x 方向上的定义域为 $0\leqslant x\leqslant L$，多数情况下 $a<<L$，因而可近似地认为 $L=Ka$，K 为整数。这样将区间[0，L]分成 K 个长度为 a 的子区间，令 $z\in[0,a]$，第 m 段子区间中的 x 值可以表示为

$$x=z+ma \qquad m=0,1,2,\cdots,K-1$$

又令 $\alpha=\frac{a}{T}$，于是有

$$f(z+ma)=\alpha g'(z+ma)+f[z+(m-1)a] \tag{3-88}$$

当 $m=0$ 时，有 $f(z)=\alpha g'(z)+f(z-a)$ 。

令 $f(z-a)=\phi(z)$ ，则

$m=0$ 时，有 $f(z)=\alpha g'(z)+\phi(z)$ ；

$m=1$ 时，有 $f(z+a)=\alpha g'(z+a)+\alpha g'(z)+\phi(z)$ 。

以此类推，将得到通式

$$f(z+ka)=\alpha\sum_{k=0}^{m}g'(z+ka)+\phi(z) \tag{3-89}$$

由于 $g'(x)$ 、α 、a 为已知，要求得 $f(x)$，只需估计出 $\phi(z)$ 即可。

式（3-89）对 $m=0,1,2,\cdots,K-1$，共 K 项累计加得

$$\sum_{m=0}^{K-1}f(z+ma)=\alpha\sum_{m=0}^{K-1}\sum_{k=0}^{m}g'(z+ka)+K\phi(z) \tag{3-90}$$

则有

$$\phi(z)=\frac{1}{K}\sum_{m=0}^{K-1}f(z+ma)-\frac{a}{K}\sum_{m=0}^{K-1}\sum_{k=0}^{m}g'(z+ka) \tag{3-91}$$

式（3-91）中右边第一项虽然未知，但是当 K 很大时，它趋于 $f(x)$ 的平均值，因此可以把第一项求和式视为一个常量 A，从而有

$$\phi(z)=A-\frac{a}{K}\sum_{m=0}^{K-1}\sum_{k=0}^{m}g'(z+ka) \tag{3-92}$$

恢复图像 $f(z+ma)$ 为

$$f(z+ma)=A-\frac{a}{K}\sum_{m=0}^{K-1}\sum_{k=0}^{m}g'(z+ka)+\alpha\sum_{k=0}^{m}g'(z+ka) \tag{3-93}$$

由于 $z+ka-ka+ma=x$，从而 $\sum_{k=0}^{m}g'(z+ka)=\sum_{k=0}^{m}g'(x-ma+ka)$ 。再利用关系

$$\sum_{k=0}^{m}g'(x-ma+ka)=\sum_{k=0}^{m}g'(x-ka)$$

最后式（3-93）可以表示成

$$f(x) = A + \alpha \sum_{k=0}^{m} g'(x-ka) - \frac{a}{K} \sum_{m=0}^{K-1} \sum_{k=0}^{m} g'(x-ka) \tag{3-94}$$

再引入去掉了的变量 y，则

$$f(x,y) = A + \alpha \sum_{k=0}^{m} g'(x-ka,y) - \frac{a}{K} \sum_{m=0}^{K-1} \sum_{k=0}^{m} g'(x-ka,y) \tag{3-95}$$

这就是去除由 x 方向上进行水平匀速直线运动造成的图像模糊后恢复图像的表达式。

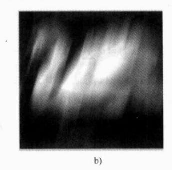
考虑到在计算机处理中，多用离散形式的公式，故而将式（3-86）和式（3-95）的离散形式写为

$$g(x,y) = \sum_{t=0}^{T} f(x - \frac{at}{T}) \Delta x \tag{3-96}$$

$$\begin{aligned} f(x,y) = {} & A + \alpha \sum_{k=0}^{m} [g(x-ka,y) - g(x-ka-\Delta x,y)] / \Delta x - \\ & \frac{a}{K} \sum_{m=0}^{K-1} \sum_{k=0}^{m} [g(x-ka,y) - g(x-ka-\Delta x,y)] / \Delta x \end{aligned} \tag{3-97}$$

上述运动模糊图像的复原处理如图 3-18 所示。


a)


b)


c)

图 3-18　图像运动模糊与去模糊实例

a) 原始图像　b) 运动造成的图像模糊　c) Wiener 滤波去图像模糊

程序代码如下：

```
%用 MATLAB 程序实现由于运动造成的图像模糊和去除图像模糊的实例
I=imread('F:\image\ swan.bmp');
figure(1);imshow(I);
%设置运动位移为 30 个像素
LEN=30;
%设置运动角度为 75°
THETA=75;
%建立二维仿真线性运动滤波器 PSF
PSF=fspecial('motion',LEN,THETA);
%用 PSF 产生退化图像
MF=imfilter(I,PSF,'circular','conv');
figure(2);imshow(MF);
```

```
%用 Wiener 滤波消除运动模糊的图像
wnr=deconvwnr(MF,PSF);
figure,imshow(wnr);
```



3.7 小结

图像在形成、传输和记录过程中，由于受多种原因的影响，图像的质量有所下降，从而引起图像的退化。图像复原是指利用退化现象的某种先验知识（即退化模型），对已经退化了的图像加以重建和复原，使复原的图像尽量接近原图像。因此，图像复原处理的关键问题是建立退化模型。在对退化图像进行复原处理时，如果对图像缺乏足够的先验知识，可利用已有的知识和经验对模糊或噪声等退化过程进行数学模型的建立及描述。图像退化过程的先验知识在图像复原技术中所起的重要作用，反映到滤波器的设计上，也就是相当于寻求点扩散函数的问题。

本章在以连续函数和离散函数两种形式介绍了图像退化的一般模型后，接着按非约束复原方法、约束复原方法和其他几种图像复原技术对图像退化的复原技术进行了介绍。

非约束复原方法仅仅要求某种优化准则为最小，不考虑其他任何约束的复原方法，常用的有非约束复原的代数方法和逆滤波复原法。所谓非约束复原的代数方法是用线性代数中的理论解决图像复原问题。通常选择最小二乘方作为优化准则的基础。逆滤波复原法也叫做反向滤波法，其主要过程是首先将要处理的数字图像从空间域转换到傅里叶频率域中，进行反向滤波后再由频率域转换到空间域，从而得到复原的图像信号。

有约束图像复原技术除了要求了解关于退化系统的传递函数之外，还需要知道某些噪声的统计特征或噪声与图像的某些相关情况。根据所了解的噪声的先验知识的不同，采用不同的约束条件，从而得到不同的图像复原技术。最常见的方法有约束的最小二乘方约束复原原法、维纳滤波法，有约束最小平方复原和去除由匀速运动引起的模糊。在最小二乘方约束复原法中，为了在数学上更容易处理，常常附加某种约束条件，形成不同的约束条件，就可得到不同类型的有约束最小二乘方图像复原方法。维纳滤波是假设图像信号可近似看成平稳随机过程的前提下，按照使输入图像和复原图像之间的均方误差达到最小的准则函数来实现图像复原的方法。

除此之外，还存在一些其他的空间图像复原方法，如几何畸变校正和盲目图像复原方法。所谓盲目图像复原法是指从观察图像中以某种方式抽出退化信息，从而找出图像复原方法。对具有加性噪声的模糊图像进行盲目图像复原的方法有两种——直接测量法和间接估计法。

尽管大多数图像整体上并不稳定，但有许多图像可以被认为是局部平稳的，另外，噪声常常会限制一幅图像的可能复原程度，特别是在空间高频段。总之，在这个领域中还有很多工作要做。

习题

3-1 画图简述退化的基本模型，并画出框图。

3-2 试写出连续退化模型，并解释何为冲激响应函数。

3-3　试写出离散退化模型。

3-4　什么是约束复原？什么是非约束复原？在什么条件下进行选择？

3-5　试描述最小二乘方复原法。

3-6　用数码相机以不同的焦距在同一位置拍摄两张图片，试比较其中透镜引起的几何畸变。

3-7　在连续线性位移不变系统的维纳滤波器中，如果假设噪声与信号的功率谱之比为 $s_n=(u,v)/s_f(u,v)=\left|H(u,v)\right|^2$，试求最佳估计值 $\hat{f}(x,y)$ 的表示式。

3-8　根据图像运动模糊应用举例，自己动手试编写图像移动不同像素和角度的模糊效果。

第 4 章 图像处理的相关操作



4.1 图像类型转换

在此将对不同图像类型的相互转换进行介绍，以使读者对图像类型有一个更加清晰的认识。

在某些图像操作中，需要对图像类型进行转换。如要对一幅索引色图像滤波，首先应将它转换成真彩色图像或者灰度图像，再利用 MATLAB 7.0 对图像的灰度进行滤波，这就是通常意义上的滤波。如果不将索引色图像进行转换，MATLAB 7.0 则对图像调色板的序号进行滤波，这没有任何意义。为此，下面将对 MATLAB 7.0 图像处理工具箱中常用的类型转换函数进行说明。

1. ind2rgb()函数

ind2rgb()函数将索引图像转换成真彩色图像，其语法格式为

- RGB= ind2rgb (X，map)

该命令将具有调色板 map 的索引色图像 X 转换成真彩色图像 RGB，实际实现时就是产生一个三维数组，然后将索引色图像对应的调色板颜色值赋予三维数组。输入图像 X 可以是 double 或 uint8 类型，输出图像 RGB 为 double 类型。

2. mat2gray()函数

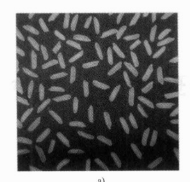
mat2gray()函数用于将一个数据矩阵转换成一幅灰度图像，其语法格式为

- I= mat2gray(A, [amin amax])
- I= mat2gray(A)

这里 I=mat2gray(A, [amin amax])按指定的取值区间[amin amax]将数据矩阵 ***A*** 转换为灰度图像 I，amin 对应灰度 0（最暗），amax 对应 1（最亮）。如果不指定区间[amin amax]，MATLAB 7.0 则自动将数据矩阵 ***A*** 中最小的元素设为 amin，最大元素设为 amax。

输入矩阵 ***A*** 和输出图像 I 都是 double 类型。实际上，mat2gray()函数与后面的 imshow()函数的功能类似。imshow()函数也可以使数据矩阵可视化。以下程序示例用 sobel 算子对图像滤波，将滤波得到的数据矩阵转换为灰度图像，转换效果如图 4-1 所示。

```
I=imread('rice.png');
J=filter2(fspecial('sobel'),I);
K=mat2gray(J);
imshow(I);
figure,
imshow(K)
```


a)

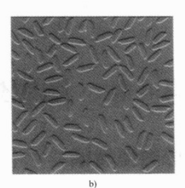

b)

图 4-1　数据矩阵转换为灰度图像

a) 原始图像　b) 转换后的图像效果

3. grayslice()函数

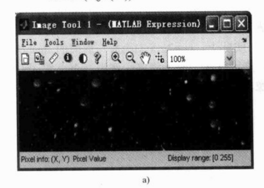
grayslice()函数通过设定阈值将灰度图像转换成索引色图像，其语法格式为

- X= grayslice(I, n)
- X= grayslice(I, v)

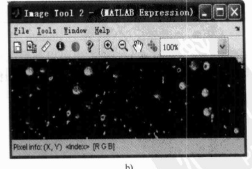
X= grayslice(I, n)命令将灰度图像 I 均匀量化为 n 个等级，然后转换为伪彩色图像 X。X =grayslice(I, v)命令则按指定的阈值向量 v（每一个元素都在 0～1 之间）对图像 I 的值域进行划分，而后转换成索引色图像 X。

输入图像 I 可以是 double 或 uint8 类型。如果阈值数量小于 256，则返回图像 X 的数据类型是 uint8，X 的值域为[0,n]或[0,length(V)]；否则，返回图像 X 为 double 类型，值域为[1,n+1]或[0,length(V)+1]。以下程序代码示例将一幅灰度图像转换成索引色图像，转换效果如图 4-2 所示。

```
I=imread('snowflakes.png');
X=grayslice(I,16);
imview(I)
imview(X,jet(16))
```



a)



b)

图 4-2　把灰度图像转换成索引色图像

a) 灰度图像　b) 索引色图像



4. rgb2gray()函数

rgb2gray()函数用于将一幅真彩色图像转换成灰度图像，其语法格式为

- I= rgb2gray(RGB)
- newmap=rgb2gray(map)

其中，I= rgb2gray(RGB)命令将真彩色图像 RGB 转换成灰度图像 I，而 newmap=rgb2gray(map)则将彩色调色板 map 转换成灰度调色板。

如果输入的是真彩色图像，图像可以是 uint8 或 double 类型，输出图像 I 与输入图像类型相同。如果输入是调色板，则输入和输出的图像都是 double 类型。

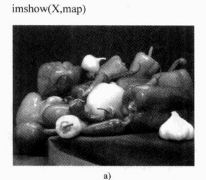
5. rgb2ind()函数

rgb2ind()函数将 RGB 图像转换成索引色图像，其调用格式为

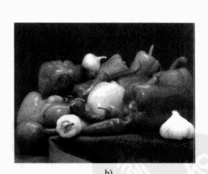
- [X, map]= rgb2ind(RGB,tol)
- [X, map]= rgb2ind(RGB, n)
- [X, map]= rgb2ind(RGB,map)
- […]=rgb2ind(...,dither_option)

支持的输入图像类型有 uint8、uint16 和 double。如果 map 的长度小于或等于 256，则输出图像的类型为 uint8。否则，输出图像类型为 uint16。以下程序代码说明了该函数的使用情况，转换效果如图 4-3 所示。

```
RGB=imread('peppers.png');
imshow(RGB);
[X,map]=rgb2ind(RGB,128);
figure;
imshow(X,map)
```



a)



b)

图 4-3 RGB 图像转换成索引色图像

a) 原始 RGB 图像 b) 索引色图像

6. im2bw()函数

im2bw()函数通过设置亮度阈值将真彩色、索引色、灰度图像转换成二值图，其语法格式为

- BW=im2bw(I,level)
- BW=im2bw(X,map,level)

- BW=im2bw(RGB,level)

以上三行命令分别将灰度图像、索引色图像和真彩色图像二值化为图像 BW。Level 是归一化的阈值，取值在[0,1]之间。输入图像可以是 double 或 uint8 类型，输出图像为 uint8 类型。以下程序代码示例对一幅图像进行二值化处理，处理结果如图 4-4 所示。

```
load trees
BW=im2bw(X,map,0.4);
imshow(X,map)
figure,imshow(BW)
```



a)



b)

图 4-4　一幅图像进行二值化处理后的结果

a) 索引色图像　b) 二值化后的图像

7. ind2gray()函数

ind2gray()函数将索引色图像转换为灰度图像，其调用格式为

- I= ind2gray(X, map)

该命令行将具有调色板 map 的索引色图像 I 转换成灰度图像 I，去掉了图像的色度和饱和度，仅保留了图像的亮度信息。输入图像可以是 double 或 uint8 类型，输入图像为 double 类型。以下程序代码将一幅索引色图像转换成灰度图像，结果如图 4-5 所示。



a)



b)

图 4-5　一幅索引色图像转换成灰度图像

a) 索引色图像　b) 转化后的灰度图像

程序代码如下：

```
load trees
I=ind2gray(X,map);
imshow(X,map)
figure,imshow(I)
```



8．dither()函数

通过抖动来转换图像，增加图像的颜色对比度。dither()函数的调用格式为

- X=dither (RGB, map)
- BW=dither (I)

这里输入图像可以是 double 或 uint8 类型。如果输出的图像是二值图像或颜色种类不超过 256 的索引色图像，则是 uint8 类型，否则为 double 类型。以下以一幅索引色图像抖动为 uint8 类型图像，程序代码如下：

```
RGB=imread('peppers.png');
[X,map]=rgb2ind(RGB,256);
I=dither(RGB,map);
BW=dither(I);
%显示索引色图像如图 4-6 所示
imshow(RGB,map);
figure;
%显示抖动转换图像如图 4-7 所示
imshow(BW)
```



图 4-6　索引色图像



图 4-7　抖动转换图像

4.2　图像数据结构

4.2.1　图像模式

1．灰度图像

灰度图像是数字图像最基本的形式，灰度图像可以由黑白照片数字得到，或对彩色图像进行去色处理得到。灰度图像只表达图像的亮度信息而没有颜色信息，因此，灰度图像的每个像素点上只包含一个量化的灰度级（即灰度值），用来表示该点的亮度水平，并且通常用 1

个字节（8 个二进制位）来存储灰度值。典型的灰度图像如图 4-8 所示。

如果灰度值用 1 个字节表示，则可以表示的正整数范围是 0～255，也就是说，像素灰度值的取值在 0～255 之间，灰度级数为 256。人眼对灰度的分辨能力通常在 20～60 级，因此，灰度值以字节为单位存储既保证了人眼的分辨能力，又符合计算机数据寻址的习惯。在特殊应用中，可能需要采用更高的灰度级数，如 CT 图像的灰度级数高达数千，需要采用 12 位或 16 位二进制位存储数据，但这类图像通常都采用专用的显示设备和软件来进行显示和处理。

2. 二值图像

二值图像是灰度图像经过二值化处理的结果，二值图像只有两个灰度级，理论上只需要 1 个二进制位来表示。在文字识别、图样识别等应用中，灰度图像一般要经过二值化处理得到二值图像，二值图像中的黑和白分别用来表示不需要进一步处理的背景和需要进一步处理的前景目标，以便对目标进行识别。图 4-9 所示为图 4-8 经过二值化处理后得到的二值图像。



图 4-8　灰度图像



图 4-9　二值图像

3. 彩色图像

彩色图像的数据不仅包含亮度信息，还要包含颜色信息。颜色的表示方法是多样化的，最常见的是三基色模型，如 RGB（Red/Green/Blue，红/绿/蓝）三基色模型，利用 RGB 三基色可以混合成任意颜色。因此，RGB 模型在各种彩色成像设备和彩色显示设备中使用，常规的彩色图像也都是用 RGB 三基色来表示的，每个像素包括红/绿/蓝三种颜色的数据，每个数据用 1 个字节（8 位二进制位）表示，则每个像素的数据为 3 个字节（即 24 位二进制位），这就是人们常说的 24 位真彩色。

4.2.2 颜色空间

1. 颜色分析

在 MATLAB 7.0 图像处理工具箱中，总是直接（RGB 图像）或者间接（索引色图像）地使用 RGB 数据来表示颜色。但是除了 RGB 颜色模型之外，还有许多其他颜色模型，如 HSV 模型等，这些不同的颜色模型就构成了所谓的颜色空间。

颜色模型是三维颜色空间中的一个可见光子集，它包括某个颜色域的所有颜色。常用的颜色模型有 NTSC（YIQ 颜色空间）、HSV 和 YCBCR 模型等。下面将主要对 RGB 颜色空间

和 YIQ 颜色空间进行简单的介绍。

（1）RGB 颜色空间简介

RGB 颜色空间是显示和保存彩色图像的最常用的彩色空间，由 R（红）、G（绿）和 B（蓝）3 个分量组成，三维空间中的 3 个轴分别与之对应。原点对应黑色，离原点最远的顶点对应白色，其他颜色落于三维空间中。由红、绿和蓝三原色组成的彩色立方体如图 4-10 所示。

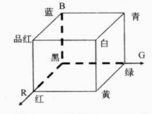


图 4-10 颜色的 RGB 空间

（2）YIQ 颜色空间简介

YIQ 颜色空间来源于国家电视标准委员会（NTSC 制式）彩色电视信号的传输，其中的 Y 分量代表图像的亮度信息，I、Q 两个分量则代表颜色信息，I 分量代表从橙色到青色的颜色变换，而 Q 分量则代表从紫色到黄绿色的颜色变化。

通过把彩色图像从 RGB 颜色空间转换到 YIQ 颜色空间，可以把彩色图像中的亮度信息与色度信息分开，从 RGB 颜色空间到 YIQ 颜色空间的转换公式为

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4-1}$$

2. 图像处理工具箱中的颜色转换函数

（1）rgb2ntsc()函数

将 RGB 颜色空间转换成 NTSC 颜色空间，使用 rgb2ntsc()函数，其调用格式为

- yiqmap=rgb2ntsc (rgbmap)
- YIQ= rgb2ntsc (RGB)

这里 yiqmap=rgb2ntsc (rgbmap)将 RGB 空间中 m×3 的色彩表 rgbmap 转换成 YIQ 颜色空间中的调色板 yiqmap。YIQ= rgb2ntsc (RGB)将真彩色图像 RGB 转换为 YIQ 颜色空间中的图像 YIQ。

（2）ntsc2rgb()函数

ntsc2rgb()函数用于将 NTSC 值转换到 RGB 颜色空间，其调用格式为

- rgbmap=ntsc2rgb (yiqmap)
- RGB=ntsc2rgb (YIQ)

（3）rgb2hsv()函数

rgb2hsv()函数用于将 RGB 模型转换成 HSV 模型，其调用格式为

- cmap=rgb2hsv(M)
- hsv_image=rgb2hsv(rgb_image)

这里 cmap=rgb2hsv(M)是将真彩色图像 M 转换为 HSV 颜色空间中的图像 HSV。hsv_image=rgb2hsv(rgb_image)将 RGB 颜色空间中 m×3 的颜色表 rgb_image 转换成 HSV 颜色空间中的调色板 hsv_image。

（4）hsv2rgb()函数

hsv2rgb()函数用于将 HSV 模型转换为 RGB 模型，其调用格式为

- M=hsv2rgb (H)



- rgb_image=hsv2rgb (hsv_image)

（5）ycbcr2rgb

ycbcr2rgb()函数用于将 YCBCR 模型转换成 RGB 模型，其调用格式为

- rgbmap=ycbcr2rgb(ycbcrmap)
- RGB=ycbcr2rgb(YCBCR)

rgbmap=ycbcr2rgb(ycbcrmap)将 YCBCR 空间中的调色板 ycbcrmap 转换成 RGB 空间中的色彩表 rgbmap。RGB=ycbcr2rgb(YCBCR)将 YCBCR 空间中的图像 YCBCR 转换为真彩色图像 RGB。

4.2.3 数据存储的数据结构

数字图像可以用矩阵来表示，因此可以采用矩阵理论和矩阵算法对数字图像进行分析和处理。最典型的例子是灰度图像，如图 4-11 所示。灰度图像的像素数据就是一个矩阵，矩阵的行对应图像的高（单位为像素），矩阵的列对应图像的宽（单位为像素），矩阵的元素对应图像的像素，矩阵元素的值就是像素的灰度值。注意：按照 C 语言的习惯，图像矩阵左上角的坐标取（0，0）。

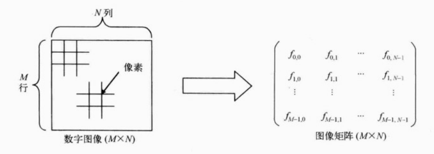


图 4-11 数字图像与图像矩阵

由于数字图像可以表示为矩阵的形式，所以在计算机数字图像处理程序中，通常用二维数组来存放图像数据，如图 4-12 所示。二维数组的行对应图像的高，二维数组的列对应图像的宽，二维数组的元素对应图像的像素，二维数组元素的值就是像素的灰度值。采用二维数组来存储数字图像，符合二维数组的行列特性，同时也便于程序的寻址操作，使计算机图像编程十分方便。

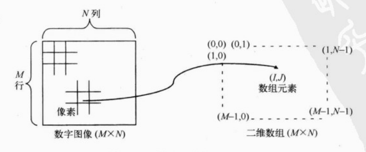


图 4-12 数字图像与二维数组

4.3 线性系统和移不变系统

工程技术中所应用的绝大多数系统在其工作范围内，其数学模型一般都可以简化为线性系统。这不仅仅是因为线性系统在理论上具有成熟的理论体系，更是因为各种应用系统（包括各种工业应用系统）在工作点附近其实际运行状态非常接近于理想的线性系统。因此，线性系统常用于描述电路系统、光学系统、机械系统、液压系统等。线性系统理论为数字信号处理、图像处理、生产自动化、信号采样与滤波以及空间分辨率的研究提供了坚实的数学基础。

4.3.1 线性系统

任何一个实际系统，当给定一个输入信号 $u(t)$ ，则产生相应的输出信号 $y(t)$ ，系统的输入信号与输出信号之间实质上是一种数学运算，可以采用如图 4-13 所示的应用系统模型表示。

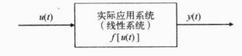


图 4-13 应用系统模型

即系统对输入信号 $u(t)$ 的作用产生了输出信号 $y(t)$ ，这种输入信号与输出信号之间的关系可以采用函数运算的形式加以描述：

$$y(t)=f[u(t)] \tag{4-2}$$

对于某一系统，若给定输入信号 $u_i(t)$ ，则产生输出信号 $y_i(t)$ ，即

$$y_i(t)=f[u_i(t)] \quad i=1,2,\cdots \tag{4-3}$$

当且仅当该系统具有如下性质时：

$$y_1(t)+y_2(t)=f[a_1u_1(t)+a_2u_2(t)]=a_1f[u_1(t)]+a_2f[u_2(t)] \tag{4-4}$$

该系统才是线性的。

上述性质表示线性系统应该满足叠加原理，即若输入信号是 N 个信号的线性加权组合，则输出信号是对上述信号中每一个信号进行同样的线性组合。叠加原理实际上包含了两个性质，即可加性和齐次性（又称为比例性）。

4.3.2 移不变系统

对于任何一个系统，若给定输入信号 $u(t)$ ，则产生输出信号 $y(t)$ ，即

$$y(t)=f[u(t)] \tag{4-5}$$

将输入信号自变量 t 沿坐标轴平移 T 时刻，若满足以下条件：

$$f[u(t-T)]=y(t-T) \tag{4-6}$$

即输出信号的函数形式不变，仅仅是输出信号的自变量平移了同样的长度 T，则称该系统具有移不变性，或称其为移不变系统。

因此，移不变系统是指系统对于输入信号 $u(t)$ 产生输出信号 $y(t)$ ，若输入信号为

$u(t-T)$ 时，则对应的输出信号为 $y(t-T)$，也就是说输入信号延时任意时刻 T，而幅值却保持不变。若线性系统满足移不变性，则称其为线性移不变系统。

4.4 调用信号分析

为了进一步研究线性系统输入、输出信号之间的规律和特性，可以从研究线性系统对复指数函数的响应着手。虽然实际系统的输入、输出信号一般都是实数，但复指数函数的实部和虚部分别是余弦信号和正弦信号，它们都是工业应用中的典型信号形式。因此，复指数信号对线性系统的研究具有重要的意义。

4.4.1 调谐信号

观察式（4-7）中的复指数信号：

$$u(t)=\cos(\omega t)+\mathrm{j}\sin(\omega t)=\mathrm{e}^{\mathrm{j}\omega t} \tag{4-7}$$

式中，$\mathrm{j}^2=1$。

函数 $u(t)$ 称为调谐信号，它是一个复函数。余弦信号和正弦信号分别是调谐信号的实部和虚部。

4.4.2 对调谐信号的响应

对于线性移不变系统，若输入调谐信号 $u_1(t)$，即

$$u_1(t)=\cos(\omega t)+\mathrm{j}\sin(\omega t)=\mathrm{e}^{\mathrm{j}\omega t} \tag{4-8}$$

则系统响应为

$$y_1(t)=H(\omega,t)\mathrm{e}^{\mathrm{j}\omega t} \tag{4-9}$$

若输入调谐信号 $u_2(t)$，即

$$u_2(t)=\cos(\omega(t-T))+\mathrm{j}\sin(\omega(t-T))=\mathrm{e}^{\mathrm{j}\omega(t-T)} \tag{4-10}$$

则系统响应为

$$y_2(t)=H(\omega,t-T)\mathrm{e}^{\mathrm{j}\omega(t-T)} \tag{4-11}$$

由于输入信号 $u_1(t)$ 和 $u_2(t)$ 存在以下关系

$$u_2(t)=\mathrm{e}^{-\mathrm{j}\omega T}u_1(t)$$

因此可得

$$\begin{aligned}\mathrm{e}^{-\mathrm{j}\omega T}y_1(t)=\mathrm{e}^{-\mathrm{j}\omega T}H(\omega,t)\mathrm{e}^{\mathrm{j}\omega T}=y_2(t)\\ y_2(t)=H(\omega,t)\mathrm{e}^{\mathrm{j}\omega(t-T)}\end{aligned} \tag{4-12}$$

比较式（4-11）和（4-12），得到

$$H(\omega,t-T)\mathrm{e}^{-\mathrm{j}\omega(t-T)}=H(\omega,t)\mathrm{e}^{\mathrm{j}\omega(t-T)} \tag{4-13}$$

即

$$H(\omega,t-T)=H(\omega,t) \tag{4-14}$$

式（4-14）中由于 T 取任意值均成立，因此，当 $H(\omega,t)$ 与 t 无关时，上式才能成立，由

此可得出

$$H(\omega,t)=H(\omega) \tag{4-15}$$

因此，可得

$$y(t)=H(\omega)u(t) \tag{4-16}$$

式（4-16）表明，线性移不变系统对于调谐信号的响应等于输入信号乘以一个函数$H(\omega)$，该函数仅仅是频率函数，即线性系统的调谐信号输入总产生同样频率的调谐信号输出。

4.4.3 系统传递函数

1．传递函数的形式

对于线性移不变系统，式（4-16）描述了输入信号与输出信号之间的关系，其中$H(\omega)$称为系统的传递函数。传递函数$H(\omega)$包含了所表示系统的全部特征。

因此，$H(\omega)$可表示为如下形式

$$H(\omega)=\frac{y(t)}{u(t)} \tag{4-17}$$

式（4-17）可写成极坐标形式

$$H(\omega)=A(\omega)\mathrm{e}^{\mathrm{j}\phi(\omega)} \tag{4-18}$$

2．线性移不变系统对余弦信号的输出

若输入为一个余弦信号，且令其为某调谐信号的实部，即

$$u(t)=\mathrm{Re}[\cos(\omega t)+\mathrm{j}\sin(\omega t)]=\mathrm{Re}(\mathrm{e}^{\mathrm{j}\omega t}) \tag{4-19}$$

则由于系统对调谐输入信号的响应为

$$H(\omega)\mathrm{e}^{\mathrm{j}\omega t}=A(\omega)\mathrm{e}^{\mathrm{j}\phi(\omega)}\mathrm{e}^{\mathrm{j}\omega t}=A(\omega)\mathrm{e}^{\mathrm{j}(\omega t+\phi)} \tag{4-20}$$

因此，系统的实际输出信号为

$$y(t)=\mathrm{Re}[A(\omega)\mathrm{e}^{\mathrm{j}(\omega t+\phi)}]=A(\omega)\cos(\omega t+\phi) \tag{4-21}$$

$A(\omega)$为系统的增益因子，代表系统对输入信号的缩放比例。ϕ为输出信号的相位，其作用是将调谐输入信号的时间坐标加以平移。

综上所述，线性移不变系统具有以下性质：

1）调谐信号输入产生同频率的调谐信号输出。

2）系统的传递函数是一个仅依赖于频率的复函数，它包含了系统的全部特征信息。

3）传递函数对调谐输入信号仅产生幅值的缩放和相位的平移。

4.5 数字图像的显示特性

一般而言，图像显示是数字图像处理的最后一个步骤，对图像所进行的各种处理完成之后，需要通过显示环节将数字图像转换为便于视觉观看的形式。

4.5.1 图像的屏幕显示

数字图像是通过光栅扫描方式和计算机内的帧缓冲存储器在计算机屏幕上进行显示的。

任何一台具有显示功能的计算机内都设有专门用于存储图像信息的帧缓冲存储器，计算机时刻自动监测该存储器，如果帧缓冲存储器内被写入图像数据，则这些数据会自动经光栅扫描方式映射到计算机显示屏幕上，从而形成数字图像。帧缓冲存储器中的每一位对应屏幕上的一个点，当某一位上的数据被置为 1 时，与该数据位对应的计算机屏幕上的相应坐标位置上就出现一个亮点，而数据位为 0 时，屏幕上的相应坐标位置则产生一个暗点。计算机开始启动时，帧缓冲存储器上所有的位被自动置为 0，当读入图像数据后，帧缓冲存储器上会根据图像数据的每一位分别置 0 或置 1。

例如，若一台计算机的显示器分辨率设定为 640 像素×480 像素，为显示一幅二值图像需要 640×480 位的帧缓存容量，这个容量称为一个位平面。因此，如果要显示一幅 256 灰度级的单色图像，则需要配置 8 个位平面，即需要 640×480 字节的帧缓存容量。若显示器设置为 800 像素×600 像素、1024 像素×768 像素或 1280 像素×1024 像素等高分辨率，则所需的帧缓存容量要求相应增大。若需要显示 R、G、B 均为 256 级的真彩色图像，则需要进行 R、G、B 三色合成，因此帧缓存的容量是单色 256 灰度级显示容量的 3 倍。

若操作系统允许用户对显示器的分辨率进行设置，则应充分考虑计算机现有的帧缓存容量的容许范围。例如，当现有帧缓存的容量为 512KB 时，为显示 256 灰度级的单色数字图像，最多只能选用 800 像素×600 像素的分辨率。如果选用 1024 像素×768 像素的分辨率，则图像的灰度级将降至 16 个等级，即 4bit 图像。可以通过增设帧缓存的容量为获取较大的分辨率，需要注意的是，若显示器不具备显示较高分辨率的能力，即使配置了足够的帧缓存，也仍然不能得到高的分辨率。

熟悉帧缓冲存储器的原理和作用之后，则可以通过直接向帧缓存填写图像数据来显示图像。需指出的是，每一个像素上的数据在帧缓存上是以位（bit）为单位描述的，而计算机中数据的输入和输出一般以字节（byte）为单位，因此图像数据中的每一个字节对应显示画面上横向排列的 8 个像素。

4.5.2 显示特性

图像需通过屏幕显示人眼才可以观察到，但图像的显示效果是由显示图像的大小、光度、分辨率、低频响应、高频响应、点间距和噪声特性等因素决定的。

1. 显示图像的大小

图像显示系统显示图像尺寸的能力包括显示器物理尺寸和系统可处理的数字图像大小两个方面。显示器自身的物理尺寸显然决定了可显示图像的大小，因此它应该尽可能大，以便观察和理解所显示的图像。显示系统能处理的最大数字图像尺寸是图像处理系统的主要指标之一，显示器物理尺寸大小不足会降低整个图像处理的效果，显示器的物理尺寸必须与待处理和显示的最大图像的行数及每行像素数相适应。

2. 光度分辨率

显示系统的光度分辨率是指系统在每个像素位置产生正确的亮度或光密度的精度。特别重要的是，显示数字图像时系统所能产生的离散灰度级数目，它部分依赖于控制每个像素亮度的位数。如果显示器只能处理 4bit 数据，则只能产生 16 种不同的灰度级。相应地，如果显示器可以处理 8bit 数据，则表示能产生 256 种灰度级。

需要指出的是，设计一个能接收 8bit 数据的显示器并不等同于可以制造一个能真正、可

靠地显示256种不同灰度级的系统。

如果显示系统内部的电子噪声达到或超过一个以上的灰度级范围时，那么显示系统灰度级的有效数目就会有一定程度的减少。一个简单的经验估计方法是根据方均根（RMS）噪声进行计算，RMS噪声级决定了灰度分辨率的实际下限。例如，若RMS噪声达到从黑到白总的显示范围的1%，则该显示器的实际灰度级可以认为只具有100级灰度的光度分辨率，即使显示系统接受8bit数据，它也只有100个有效灰度级。因此，有效灰度级数绝对不会多于数字数据中的灰度级数，而很可能会少一些。



3. 灰度的线性特性

灰度的线性特性是一个非常重要的显示特性指标，它的含义是显示图像的亮度或密度正比于输入灰度级的程度。任何显示设备都有一条输入灰度级与输出亮度的变换曲线。为了进行正确的运算操作，这条曲线应当是线性的，并且保持恒定。对包含（需经显影及放大的）胶片记录器的永久性显示设备来说，必须有精细的质量控制方能得到再现的结果。

需要指出的是，人眼并不是精确的光度计，如果转换曲线中存在轻度的非线性，以及从图像的一侧到另一侧10%～20%的亮度阴影渐变是很难被视觉所捕获的。但如果变换曲线的两端有明确肩部（或趾部），则在亮区（或暗区）可能会丢失信息或导致图像质量下降。

4. 显示标定

在使用电视监视器进行暂时性显示时，变换曲线部分依赖于监视器的亮度和对比度旋钮的调整位置。硬拷贝印制机一般在前面板或后面板上有一个以上的调节钮。有时，它们包括一个用来调整非线性变换曲线形状的设定旋钮。这样就让用户能修改非线性变换曲线以适合他们的特殊要求或个人习惯。在大多数情况下，这些处理应该是由软件而不是由显示系统来完成的——显示系统仅起把数据无附加“增强”地显示给操作者的作用。

显示标定能保证所显示的图像正确地表达该数字图像。一幅包含所有灰度级的线条（或正方形）灰度测试卡被显示在监视器上或送往图像记录器，然后调整不同的设置以使全部亮度范围都可见，并且在两端均没有灰度级的丢失。一个图像处理系统经过合理的显示标定后，从硬拷贝记录器印制出的结果就与屏幕上显示的图像看起来正好一样，而且它也是数字图像数据的精确表达。

5. 低频响应

低频响应本质上是指显示系统再现大块等灰度级区域（平坦区域）的能力。数字图像处理技术的目标是使数字处理对图像的视觉效果影响最小，也就是说，希望平坦区域以均匀一致的亮度显示出来。

假设显示亮点具有如下形式的二维高斯分布：

$$p(x,y)=\mathrm{e}^{-(x^2+y^2)}=\mathrm{e}^{-r^2} \tag{4-22}$$

式中，r 是从亮点中心量起的径向距离。如果定义 R 为亮度等于最大值一半处的半径，则可以将点分布曲线函数写为如下形式：

$$p(r)=\mathrm{e}^{-(r/R)^2\ln 2} \tag{4-23}$$

上式可以进一步简化为

$$p(r)=2^{-(r/R)^2} \tag{4-24}$$

因此，亮度分布如图4-14所示。根据以上推导，单个点的亮度分布可表示如下：

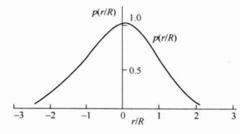


图 4-14　高斯点的高度剖面

$$p(x,y)=2^{-[(x^2+y^2)/R^2]} \tag{4-25}$$

由于高斯点只有在离中心大约两倍以上半径的距离时亮度才降至不足其峰值的 1%，因此除非显示点间距较大才不会发生重叠，距离太小则会发生重叠。注意，点中心与两点中间位置的亮度差为 12.5%，因此，在 $d=2R$ 时无法给出微观平坦的区域。

6. 高频响应

对于图像显示特性而言，高频响应是指系统再现直线图案性能的好坏，系统再现直线图案的性能反映了显示图像细节的能力。显示一种常用的高频测试图案由相距一个像素的明暗交替竖直线构成，有时被称为“线对”，其中每一线对包括一条暗线（由零亮度像素组成）和相邻的一条亮线（由高亮度像素组成）。

如图 4-15 所示，在高频竖线图案中，粗黑点代表高亮度的像素，小黑点代表零亮度的像素。因此，可以得出位于线上的像素中心的亮度为

$$D(0,0)\approx 1+2p(d)+2p(2d) \tag{4-26}$$

而线间像素亮度可以用下式表示：

$$D(1,0)\approx 2p(d)+4p(\sqrt{2}d) \tag{4-27}$$

调制系数 M 可用下式表示：

$$M=\frac{D(0,0)-D(1,0)}{D(0,0)} \tag{4-28}$$

M 与点间距的关系如图 4-16 所示。根据该曲线，当点间距小于 $2R$ 时，调制深度迅速下降。

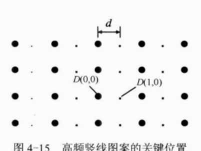


图 4-15　高频竖线图案的关键位置

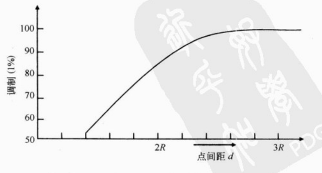


图 4-16　间距对高频竖线图案的影响

7．点间距的选择

显示点间距越小，均匀区域的平坦性越好；而间距越大，则越能更好地再现图像细节及其对比度。因此，就点间距的选择而言，区域平坦性与高频响应的要求是一对矛盾。实际选择时须采取折中方法，具体的折中方案取决于实际图像中高频和低频信息的相对重要性。点间距可认为是一个在图像处理中必须适当调整的显示参数。

8．噪声考虑

显示系统的电子噪声可能引起的不利包括显示点位置与亮度两个方面。

（1）位置噪声

一种较严重的影响来自偏转电路，即可能引起点显示间距的不均匀。除非极其严重，否则显示位置噪声不会给图像带来视觉系统可观察到的几何畸变。然而，点相互影响与位置噪声的组合会产生较大的幅值变化，因为点相互影响效应放大了位置噪声，要得到好的显示特性必须精确控制像素的位置。

（2）幅值噪声

如果所有噪声（包括随机噪声和周期性噪声）幅值都低于一个灰度级，那么总的显示效果较理想。亮度通道的随机噪声会产生一种“胡椒加盐”（即黑白噪声点）的效果，在平坦区域中，尤其明显。而对于与水平或垂直偏转信号相同的周期性噪声，它可能产生条状图案。

4.5.3 数字图像的暂时显示

最常见的暂时显示是采用光栅扫描的阴极射线管（CRT）和液晶显示器（LCD）。

对于 CRT，其像素点的亮度随着位置的不同而变化，从而产生图像，显示系统根据存储的数字图像数据，周期性地连续扫描和刷新显示点。对于普通的电视监视器若能提供合适的视频信号，也可用做数字图像显示器。

LCD 是另一种常用的显示设备，它是一种低电压、低功耗器件，可直接由 MOS-IC 驱动，因此器件和驱动系统之间的配合较好。LCD 的优点是结构简单，平面型，显示面也可任意加工，而且 LCD 是反射型的，在室内也很容易看到数字图像。目前，台式计算机、便携式计算机、手表、玩具、计算器、民航候机牌、火车候车室、户外广告，以及测量仪器和航天测控显示等都广泛使用了液晶显示设备。

4.5.4 数字图像的永久显示

永久显示是指将图像永久记录在纸或胶片等介质上的过程，其中图像记录器、打印机和其他图像硬拷贝设备是常用的图像永久显示设备。

1．抖动技术和彩色印制

（1）抖动技术

抖动技术是弥补所用颜色不足的一个有效办法，即以实心点图案仿真灰度级的处理过程称为抖动或半影调。抖动技术有多种方式，但不论采用什么方式，所有抖动技术的基本原理相同，即用直接显示其色彩的像素模式来替换那些色彩不能直接显示的像素。抖动技术利用了生活中的一个简单原理：人的肉眼能将两种不同颜色的相邻像素融合成第三种颜色。一种抖动方法可能用一个蓝色像素与黄色像素交错的模式去替换一块绿色像素，这种处理方法称

为模式抖动。模式抖动存在的问题是有时成组的非相关像素会结合形成被称为人工痕迹的子模式，它们能影响最终的图像。抖动图像的一个更好办法是使用一种叫扩散抖动的技术，它不依赖于事先设定的颜色模式，而是着眼于图像中的每一个像素，并给它指定一种尽可能与其原来颜色匹配的新颜色，然后通过适当的计算以量化新旧颜色之间的差异，并将这种差异扩散到其相邻像素的颜色中。例如，如果一个像素的新颜色中所包含的红色与绿色比旧颜色中所包含的红色与绿色少，则扩散抖动会给周围的像素中增加适当的红色与绿色。这种对抖动的适应处理清除了人工痕迹并且通常能产生更好的效果。抖动也能用于为打印机这类单色设备产生彩色图像的黑白再现，如半影调处理过程可用于产生报纸上的黑白图片。

（2）彩色印制

人类的视觉系统能独立地感觉红（Red），绿（Green），蓝（Blue）三个波段光，人眼能识别的颜色均可由红、绿、蓝通过适当混合来实现，因此它们又称为三基色。例如，彩色 CRT 就是利用了这个特性。

硬拷贝的视觉效果是通过反射光获得的，其图像的基本构成元素是三种颜料，分别吸收红光、绿光和蓝光。在白光下，吸收红光的颜料呈现青色，吸收绿光的颜料呈现品红色，吸收蓝光的颜料呈现黄色。理论上，按不同比例混合这三种颜料可以获得任何视觉可见的颜色。青色（Cyan）、品红（Magenta）、黄色（Yellow）称为补色，被用于彩色打印中的三个基本颜色，这就是所谓的 CMY 系统。

理论上，若将等量青色、品红和黄色的颜料混合，由于红、绿、蓝光被它们分别吸收了，因而呈现黑色，混合三种补色而得到的黑色被称为合成黑。而稀释的混合物，由于只能吸收一部分入射光而呈现出灰色。实际应用中，由于不能产生合适的灰色影调，因此，实际的彩色印刷通常使用第四种墨水——黑（Black）墨水，以保证正确表达灰度信息，这种系统称为 CMYK 系统或四色印刷法。

2．永久记录设备

（1）喷墨打印机

喷墨打印机的输出图像分辨率通常在 300～600dpi 之间。无论是单色还是彩色液体喷墨打印机，都是通过一组微小喷嘴（安装在可随意组装的夹头上）喷射出墨水流，由于墨水的不透明性，因此，需要使用抖动技术来产生不同的灰度。墨水喷到纸或胶片上，干爽后形成一个个小点，必要时可通过加热来缩短干燥过程，防止墨水在干燥过程中发生浸润。

使用普通纸打印时，墨水干燥之前会在纸上扩散，使打印的像素略有变大，使图像变得模糊。若采用专用纸，则可以较好地控制这种现象，使图像更清晰。

（2）激光打印机

激光打印机是目前应用最广泛的高质量打印机，其优点是打印质量高，分辨率高，色彩艳丽，噪声低，速度快，不足之处是价格较贵，打印成本高，不能打印多层介质。

激光打印机是通过电子成像技术进行打印的，当调制激光束在硒鼓上沿轴向扫描时，根据字符信息的点阵结构，使硒鼓面感光，形成负电荷阴影，在硒鼓面经过带正电荷的墨粉时，感光部分会吸附上墨粉，并将墨粉转印到纸上，纸上的墨粉经加热熔化形成永久性的字符与图像。

（3）图像记录器

一种常用的图像记录器是 CRT 胶片记录器。CRT 胶片记录器实质上是一个安装在 CRT

显示器前的照相机，当快门打开时，整个图像像素逐个显示以使胶片曝光。它只需要显示一遍，无需刷新。像素亮度的调制有两种方法，可以通过控制显示点的亮度来实现；还可以通过控制每个像素的显示时间来实现。胶片上像素点的曝光量与曝光强度和曝光时间的乘积成比例。

（4）静电绘图仪

静电绘图仪是一种利用静电同极相斥、异极相吸的原理进行工作的光栅扫描设备。单色静电绘图仪是把像素化后的绘图数据输出至静电写头上，静电写头一般是双行排列，写头内装有许多电极针。写头根据输入信号控制每根电极针所放出的高电压，绘图纸正好横跨在写头与背板电极之间，纸张通过写头时，写头便将图像信号转换到纸上。带电的绘图纸经过墨水槽时，由于墨水的碳微粒带正电，所以墨水被纸上的电子吸附，从而形成图像。彩色静电绘图仪的原理与单色静电绘图仪的原理类似，根据 CMYK 系统的颜色合成理论，当进行彩色绘图时，纸张往返运动，分别套上品红、黄、青、黑四种颜色，根据比例可形成各种所需要的彩色。

目前，彩色静电绘图仪的分辨率可达 800dpi 或更高，若选用高质量的墨水和纸张，则所产生的彩色图片比彩色照片的质量更好。

（5）热蜡转移打印机

热蜡转移打印机使用塑料胶片，塑料胶片上的涂层是浸过颜料的蜡。这种色带带有矩形的面板，每个面板与打印样张大小相同，并涂以墨水。面板上的颜色有青色、品红、黄色和黑色。与喷墨打印机的原理类似，热蜡转移打印机也需要采用抖动技术形成彩色影调。工作时，每次选用 CMYK 颜色系统中的一种颜色，具体方法即将相应的色带面板与打印介质接触并由打印头进行绘制。打印头上有成千上万的微小加热元件，可以按控制信息通电使蜡熔化并转移到纸上产生一个点，打印分辨率可达 300dpi 以上。

4.6 二维系统及矩阵运算

对于数字图像处理技术而言，二维线性系统具有重要的意义和广泛的应用，同时也是学习其他章节的基础。

4.6.1 二维线性系统

设 $f_i(x,y)$ 是二维系统的给定输入，$g_i(x,y)$ 表示其输出，即

$$g_i(x,y)=T[f_i(x,y)] \quad i=1,2,\cdots \tag{4-29}$$

若该系统的输入、输出满足以下特性：

$$\begin{aligned}a_1g_1(x,y)+a_2g_2(x,y)&=a_1T[(f_1(x,y)]+a_2T[(f_2(x,y)]\\&=T[a_1f_1(x,y)+a_2f_2(x,y)]\end{aligned} \tag{4-30}$$

则称该系统为二维线性系统。

4.6.2 二维位置不变线性系统

对于任意一个二维系统，若给定输入 $f(x,y)$，则产生输出 $g(x,y)$，即

$$g(x,y)=T[f(x,y)] \tag{4-31}$$

将输入信号自变量x和y分别平移x_0和y_0，若满足以下条件：

$$g(x-x_0,y-y_0)=T[f(x-x_0,y-y_0)] \tag{4-32}$$

即输出信号的函数形式不变，仅仅是输出信号的自变量分别平移同样长度x_0和y_0，则二维系统具有移不变特性，或称为二维移不变系统，又称为二维位置不变系统。若二维系统既满足线性特性，又满足位置移不变特性，则称为二维位置不变线性系统（也可称为空间不变线性系统）。

二维位置不变线性系统的输出也可以通过输入信号与其冲激响应函数$h(x,y)$的二维卷积求出。对于二维连续系统，其形式如下：

$$\begin{aligned}g(x,y)&=f(x,y)h(x,y)\\&=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(u,v)h(x-u,y-v)\mathrm{d}u\mathrm{d}v\end{aligned} \tag{4-33}$$

对于二维离散系统，其形式如下：

$$g(i,j)=\sum_m\sum_n f(m,n)h(i-m,j-n) \tag{4-34}$$

4.6.3 二维系统的梯度算子

数字图像处理及分析中，无论是图像的增强处理、图像的复原，还是边缘检测，许多处理方法需要应用二维系统的梯度算子，梯度是图像处理算法中的一个重要概念。

1. 连续系统梯度算子

对于连续系统，在坐标位置(x,y)处的梯度向量为

$$i\frac{\partial}{\partial x}f(x,y)+j\frac{\partial}{\partial y}f(x,y) \tag{4-35}$$

上式也可写为

$$i\frac{\partial}{\partial x}+j\frac{\partial}{\partial y} \tag{4-36}$$

由于梯度是向量，因此其幅值为

$$\sqrt{\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2} \tag{4-37}$$

梯度的方向为

$$\theta=\arctan\left(\frac{\partial f}{\partial y}\Big/\frac{\partial f}{\partial x}\right) \tag{4-38}$$

2. 离散系统梯度算子

在数字图像处理中，包括常用的罗伯特算子、索贝尔算子、普瑞维特算子等在内的各种梯度算子均以差分形式表示。因此，差分梯度算子在图像处理中具有广泛的应用领域。由于无论是x方向还是y方向，离散系统坐标值的最小增量为1，因而以相邻点之差近似表示梯度分量：

$$\begin{cases}\Delta_x f=f(m,n)-f(m-1,n)\\\Delta_y f=f(m,n)-f(m,n-1)\end{cases} \tag{4-39}$$

梯度的幅值为

$$\nabla f=\sqrt{(\Delta_x f)^2+(\Delta_x f)^2} \tag{4-40}$$

在某些情况下，为简单起见（避免平方根运算），可以根据具体情况分别采用如下方法计梯度的近似值：

$$\nabla f\approx \max(|\Delta_x f|,|\Delta_y f|) \tag{4-41}$$

显然，离散系统梯度的幅值与上述两种近似值之间存在以下关系：

$$\max(|\Delta_x f|,|\Delta_y f|)\leqslant\sqrt{(\Delta_x f)^2+(\Delta_y f)^2}\leqslant|\Delta_x f|+|\Delta_y f| \tag{4-42}$$

4.6.4 常用矩阵运算

1. 矩阵的转置与共轭

设 $\boldsymbol{A}$ 表示 $M\times N$ 矩阵，矩阵 $\boldsymbol{A}$ 表示如下：

$$\boldsymbol{A}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ a_{31} & a_{32} & \cdots & a_{3N} \\ \vdots & \vdots & & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{pmatrix} \tag{4-43}$$

式（4-43）简单表示为 $\boldsymbol{A}=\{a_{ij}\}$。

式中，$a(i,j)$ 表示矩阵 $\boldsymbol{A}$ 的第 i 行第 j 列元素。i 与 j 的取值分别如下：

$$i=1,2,3,\cdots,M$$
$$j=1,2,3\cdots,N$$

则矩阵 $\boldsymbol{A}$ 的转置矩阵如下：

$$\boldsymbol{A}^{\mathrm{T}}=\begin{pmatrix} a_{11} & a_{21} & \cdots & a_{M1} \\ a_{12} & a_{22} & \cdots & a_{M2} \\ a_{13} & a_{23} & \cdots & a_{M3} \\ \vdots & \vdots & & \vdots \\ a_{1N} & a_{2N} & \cdots & a_{MN} \end{pmatrix}=\{a_{ji}\} \tag{4-44}$$

式中，i 与 j 的取值范围不变。

矩阵 $\boldsymbol{A}$ 的共轭矩阵如下：

$$\boldsymbol{A}^*=\begin{pmatrix} a_{11}^* & a_{12}^* & \cdots & a_{1N}^* \\ a_{21}^* & a_{22}^* & \cdots & a_{2N}^* \\ a_{31}^* & a_{32}^* & \cdots & a_{3N}^* \\ \vdots & \vdots & & \vdots \\ a_{M1}^* & a_{M2}^* & \cdots & a_{MN}^* \end{pmatrix} \tag{4-45}$$

即

$$\boldsymbol{A}^*=\{a_{ij}^*\} \qquad i=1,2,3,\cdots,M;\ j=1,2,3,\cdots,N$$

同理，可以得出其共轭转置矩阵为

$$(\boldsymbol{A}^*)^{\mathrm{T}} = \{a_{ji}^*\} \qquad i=1,2,3,\cdots,M; j=1,2,3,\cdots,N \tag{4-46}$$

2. 矩阵的行列式与逆矩阵

虽然矩阵与行列式在形式上类似，它们都是以矩形陈列，但却是两个不同的概念。行列式是一个数，而是指按一定的运算规律所确定的一个数，它要求行数与列数必须相同。矩阵不是一个数，而是按一定方式排列的一张有序的数值表，其行数与列数可以不同。当矩阵的行数与列数相同时，则称其为方阵，由 n 阶方阵 $\boldsymbol{A}$ 的元素构成的 n 阶行列式，称为方阵 $\boldsymbol{A}$ 的行列式，记为$|\boldsymbol{A}|$，或 $\det \boldsymbol{A}$ 。

如果方阵 $\boldsymbol{A}$ 的行列式$|\boldsymbol{A}| \neq 0$，则称 $\boldsymbol{A}$ 可逆，其逆矩阵 $\boldsymbol{A}^{-1}$ 满足下式：

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$$

n 阶方阵具有以下主要性质：

1）$|\boldsymbol{A}| = |\boldsymbol{A}^{\mathrm{T}}|$

2）$|c\boldsymbol{A}| = c^n|\boldsymbol{A}|$

3）$|\boldsymbol{AB}| = |\boldsymbol{BA}| = |\boldsymbol{A}||\boldsymbol{B}|$

4）$(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$

5）$|\boldsymbol{A}^{-1}| = 1/|\boldsymbol{A}|$

6）$(\boldsymbol{A}^{\mathrm{T}})^{-1} = (\boldsymbol{A}^{-1})^{\mathrm{T}}$

如果 $\boldsymbol{A}$ 是一个 $M \times N$ 矩阵，且$(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}$存在，则它的伪逆用 $\boldsymbol{A}^{-}$ 表示，计算形式如下：

$$\boldsymbol{A}^{-} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}} \tag{4-47}$$

并且下式成立：

$$\boldsymbol{A}^{-}\boldsymbol{A} = \boldsymbol{I}$$

3. 正交矩阵和酉矩阵

正交矩阵和酉矩阵都是数字图像处理运算中经常应用的概念。

（1）正交矩阵

如果矩阵 $\boldsymbol{A}$ 满足：

$$\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}} \tag{4-48}$$

则称矩阵 $\boldsymbol{A}$ 为正交矩阵。

（2）酉矩阵

对于复数矩阵，如果矩阵 $\boldsymbol{A}$ 满足：

$$\boldsymbol{A}^{-1} = (\boldsymbol{A}^*)^{\mathrm{T}} \tag{4-49}$$

则称矩阵 $\boldsymbol{A}$ 为酉矩阵。

4. 矩阵的迹

设 λ 为一个数，对于 n 阶方阵 $\boldsymbol{A}$，如果存在 n 维的非零列向量 v，使下式成立

$$\boldsymbol{A}v = \lambda v \tag{4-50}$$

则称 λ 为方阵 $\boldsymbol{A}$ 的特征值，v 为方阵 $\boldsymbol{A}$ 对应于特征值 λ 的特征向量。

方阵 $\boldsymbol{A}$ 的特征值 λ 可以通过求解代数方程求解：

$$|\boldsymbol{A} - \lambda\boldsymbol{I}| = 0 \tag{4-51}$$

n 阶方阵 $\boldsymbol{A}$ 共有 n 个特征值，且不同的特征值所对应的特征向量线性无关。

n 阶方阵 $\boldsymbol{A}$ 主对角线上的 n 个元素之和称为方阵 $\boldsymbol{A}$ 的迹，用 $\mathrm{tr}(\boldsymbol{A})$ 表示方阵的迹，即

$$\mathrm{tr}(\boldsymbol{A})=\sum_{i=1}^{n}a_{ii} \tag{4-52}$$

方阵的迹具有以下性质：

1） $\mathrm{tr}(\boldsymbol{A}+\boldsymbol{B})=\mathrm{tr}(\boldsymbol{A})+\mathrm{tr}(\boldsymbol{B})$

2） $\mathrm{tr}(c\boldsymbol{A})=c\,\mathrm{tr}(\boldsymbol{A})$

3） $\mathrm{tr}(\boldsymbol{AB})=\mathrm{tr}(\boldsymbol{BA})$

4）若 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 为方阵 $\boldsymbol{A}$ 的 n 个特征值，则有

$$\mathrm{tr}(\boldsymbol{A})=\lambda_1+\lambda_2+\cdots+\lambda_n$$

5） $|\boldsymbol{A}|=\lambda_1\lambda_2\cdots\lambda_n$

4.7 图像的块操作

在 MATLAB 7.0 图像处理中，有时并不需要对整个图像进行操作，而是对图像中的某一部分（即块）进行操作。例如，许多线性滤波操作和二进制图像操作均按照块操作方式实现。MATLAB 7.0 的图像处理工具箱提供了多个专门用于图像块操作的函数，如 dilate()函数等。此外，工具箱提供的大量普通函数也适用于块操作。利用这些函数，用户可以进行各种块操作，包括边缘操作和显示块操作等。

4.7.1 边缘操作

边缘操作属于块操作，一次只处理一个图像像素。在 MATLAB 7.0 中，像素的边缘是一个像素集，由该像素（通常称为中心像素）的相关位置确定。从本质上讲，像素的边缘就是由像素集构成的一个矩形块。当操作从图像矩阵的一个元素移动到另一个元素时，该矩形块也以相向的方向同时移动，如图 4-17 所示。

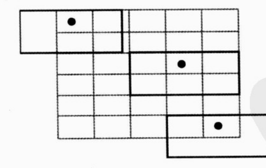


图 4-17　边缘的移动

图 4-17 中的边缘为一个 2×3 的矩形块，黑点表示边缘的中心像素。通常，对于 m×n 的边缘来说，中心像素的计算方法如下：

```
floor ( [ ( m+1 n+1) ] /2 )
```

例如，图 4-17 中边缘的中心像素为(1,2)，即边缘块中的第一行第二列对应的元素。

MATLAB 7.0 进行边缘操作的过程如下：

1）选择像素。

2）确定该像素的边缘块。

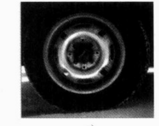
3）调用适当的函数对边缘中的元素进行操作。

4）查找对应于输出图像中心像素的像素点。

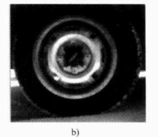
5）对于输入图像中的每个像素点，重复 1）～4）的操作。

以下程序代码示例即调用 nlfilter()函数进行边缘操作的过程，其操作效果如图 4-18 所示。

```
B=imread('tire.tif');
g=inline('max(x(:))');
B2=nlfilter(B,[3 3],g);
figure,imshow(B);
figure,imshow(B2)
```



a)



b)

图 4-18　边缘操作前、后的效果比较

a) 原始图像　b) 边缘操作后的图像

4.7.2 显示块操作

MATLAB 7.0 中的显示块是将矩阵划分为 $m \times n$ 后得到的矩形部分，如图 4-19 所示。

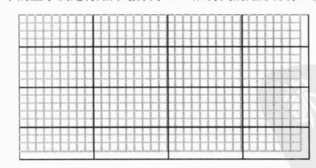


图 4-19　显示块

图中的虚灰色方格为数据矩阵，而黑色的矩形则为显示块。

MATLAB 7.0 进行显示块操作的函数为 blkproc()函数，该函数能够将每个显示块从图像中提取出来，然后将其作为参数传递给任何用户函数（即用户指定的函数）。此外，bklproc()函数还可以对由用户函数返回的显示块进行组合，从而生成最后的输出图像。

以下命令行利用 4×6 的显示块处理矩阵 I，其中用户函数为 myfun()。

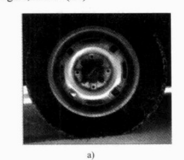
```
B2=blkproc(B,[4 6],'myfun');
```

另外，还可以把用户函数指定为一个嵌入式函数（即 inline()函数）。在这种情况下，出现在 blkproc()函数中的嵌入式函数不能带有任何引用标记，程序代码如下：

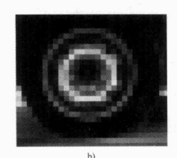
```
g=inline('mean2(x)*ones(size(x)) ');
B2=blkproc(B,[4 6],g);
```

以下程序代码示例利用 blkproc()函数，将图像 tire.tif 数据矩阵中的每个 8×8 显示块中的每个像素值设置为该显示块中的平均值，操作效果如图 4-20 所示。

```
B=imread('tire.tif');
g=inline('uint8(round(mean2(x)*ones(size(x))))');
B2=blkproc(B,[8 8],g);
figure,imshow(B);
figure,imshow(B2)
```



a)



b)

图 4-20　显示块操作前、后的效果比较

a) 操作前　b) 操作后

最后需要说明的是，在调用 blkproc()函数定义显示块时，可以将其指定为彼此交叠的显示块，如图 4-21 所示。定义交叠显示块的程序代码如下：

```
B=blkproc(A , [4 8] , [1 2] , @myfun)
```

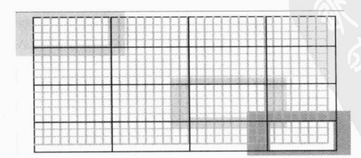
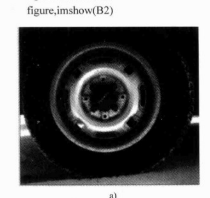


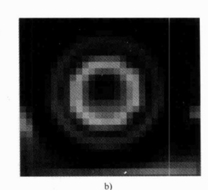
图 4-21　彼此交叠的显示块

如果将图 4-20a 所示图像中的显示块定义为具有 4×4 的交叠显示块，则处理效果如图 4-22 所示。

```
B=imread('tire.tif');
g=inline('uint8(round(mean2(x)*ones(size(x))))');
B2=blkproc(B,[8 8],[4 4],g);
figure,imshow(B);
figure,imshow(B2)
```



a)



b)

图 4-22　显示块交叠前、后的显示效果

a) 交叠前　b) 交叠后

4.8　特定区域处理

4.8.1　特定区域

在进行图像处理时，有时只需对图像中的某个特定区域进行处理，并不需要对整个图像进行处理，如要对用户选定的一个区域进行均值滤波或对比度增强。MATLAB 7.0 图像处理工具箱提供的某些处理函数可以对指定的区域进行处理。

对特定区域的处理通过二值掩模来实现。当用户选定一个区域后，会生成一个与原图像大小相同的二值图像，选定的区域为白色，其余部分为黑色，这样就可以实现对特定区域的选择性处理了。

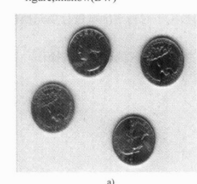
MATLAB 7.0 图像处理工具箱提供了 4 个函数分别支持对特定区域的选择、滤波和填充。

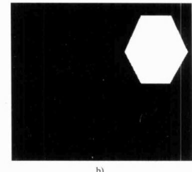
roipoly()函数用于选择图像中的多边形区域，其调用格式参考 MATLAB 7.0 图像处理工具箱。

以下程序代码说明了 roipoly()函数的用法，并根据指出的坐标选择一个六边形区域，其处理效果如图 4-23 所示。

```
B=imread('eight.tif');
g=[222 272 300 270 221 194];
f=[21 21 75 121 121 75];
BW=roipoly(B,g,f);
```

```
figure,imshow(B);
figure,imshow(BW)
```







a)　　b)

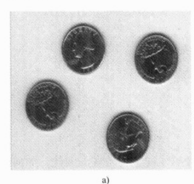
图 4-23　根据特定的坐标选择六边形

a) 原始图像　b) 选择后的图像

4.8.2 特定区域滤波

MATLAB 7.0 图像处理工具箱中提供了 roifilt2()函数用于对一个区域滤波，其调用格式参考 MATLAB 7.0 图像处理工具箱。

以下程序代码示例说明了 roifilt2()函数的用法，对特定区域进行锐化滤波，处理结果如图 4-24 所示。





a)　　b)

图 4-24　对特定区域进行滤波的结果

a) 原始图像　b) 滤波后的图像

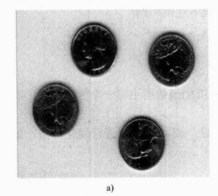
```
B=imread('eight.tif');
g=[222 272 300 270 221 194];
f=[21 21 75 121 121 75];
BW=roipoly(B,g,f);
h=fspecial('unsharp');
j=roifilt2(h,B,BW);
figure,imshow(B);
figure,imshow(j)
```

4.8.3 特定区域填充

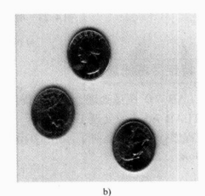
MATLAB 7.0 图像处理工具箱中提供了 roifill()函数用于对待定区域进行填充操作，其调用格式可参考 MATLAB 7.0 图像处理工具箱。

以下程序代码示例说明了 roifiltl()函数的用法，并对特定区域进行填充操作，处理结果如图 4-25 所示。

```
B=imread('eight.tif');
g=[222 272 300 270 221 194];
f=[21 21 75 121 121 75];
j=roifill(B,g,f);
figure,imshow(B);
figure,imshow(j)
```



a)



b)

图 4-25 对特定区域进行填充的结果

a) 原始图像 b) 填充后的图像

4.9 图像质量评价

绝大多数情况下，图像处理的目的是为了改善图像的（视觉）质量，因此，如何评价图像的质量是一个十分重要的问题。例如，图像增强的处理目的是以各种可能的形式突出图像中用户感兴趣的区域，抵制图像中的随机噪声，提高图像的视觉质量。图像复原的处理目的是建立图像质量退化的数学模型，对图像质量退化进行相应的补偿，包括运动模糊补偿、焦距模糊补偿以及噪声消除等。图像编码压缩的目的则是在尽可能保持图像质量（无损或有损压缩）的条件下，对图像数据进行编码压缩以减少数据存储量和传输量。这些处理都涉及到图像质量评价问题。

由于人类视觉及其系统的高度复杂性，图像质量评价事实上一直是一个十分困难的问题，可以说迄今为止，还没有一种权威、系统并得到公认的、一应俱全的体系和评价方法。目前，常见的图像质量评价方法分为客观评价和主观评价两种。

4.9.1 图像质量的客观评价

图像质量的客观评价是指提出某个或某些定量参数和指标来描述图像质量。例如，在图像压缩时，评价图像质量的定量参数可以选用解压图像对基准图的误差参数，如常见的定量参数是方均误差 *MSE* 和峰值信噪比 *PSNR*。

$$MSE = \frac{1}{M \times N} \sum_x \sum_y (f_x(x,y) - f(x,y))^2 \tag{4-53}$$

$$PSNR = 10 \times \lg \frac{(f_{\max} - f_{\min})^2}{MSE} \tag{4-54}$$

式中，M、N 分别对应图像的列数和行数；$f(x,y)$、$f_x(x,y)$ 分别为原始图像和解压重建的图像；$f_{\max}$、$f_{\min}$ 分别对应图像灰度的最大值和最小值（通常取 255 和 0）。



图像质量的客观评价的另一种方法是采用测试卡。在测定电视的显示质量、数码相机和扫描仪的成像质量时，常用不同的标准测试卡来完成，如在测定数码相机的分辨率时，通常用专业的标准分辨率测试卡（见图 4-26）进行照相，然后利用配套软件对标准测试卡图像进行观察和计算，可以测出数码相机的分辨率（线数）。

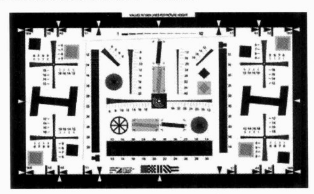


图 4-26 ISO 12233 标准分辨率测试卡

客观评价的特点是采用客观指标和定量指标，评价结果原则上不受人为干预和影响，但由于目前的定量参数还不能或者不完全能反映人类视觉的本质，对图像质量的客观评价指标经常与视觉的评价有偏差，甚至有时结论完全相反。

4.9.2 图像质量的主观评价

图像质量的主观评价是指采用目视观察和主观感觉评价图像的质量。

主观评价的方法类似于体操比赛的评分，由数名裁判组成评分小组，根据规则要求和评分标准对体操运动员的比赛动作进行打分，评分结果取总和或平均值。有些比赛还采取去掉最高分和最低分的方法，以减少带有倾向性打分的影响。

图像质量主观评价的“裁判”可以由未经训练的普通观察者来担任，或由专业图像判读员和图像专家来担任。也可由未经训练的普通观察者和专业图像判读员分组（普通观察者和专家组）进行评价。评价时需要事先制定评分标准以及评分规则，然后依据标准和规则进行

分组评价，世界各主要国家和组织对图像质量主观评价的评分标准见表 4-1。

图像质量主观评价的特点是主观性和定性评，评价结果受人为影响和干扰较多，但由于目前的图像质量客观评价的指标和参数尚不能完全反映主观视觉对图像质量的评价，所以图像质量主观评价还是最重要的评价方法之一。

表 4-1　世界各主要国家和组织对图像质量主观评价的评分标准

损伤			质量			比较		
每级的主观质量		国家和组织	每级的主观质量		国家和组织	比较的衡量		国家和组织
五级标准	5 不能察觉 4 刚察觉不讨厌 3 有点讨厌 2 很讨厌 1 不能用	原联邦德国、日本等	五级标准	5 优 4 良 3 中 2 次 1 劣	原联邦德国、日本、英国	五级标准	+2 好得多 +1 好 0 相同 -1 坏 -2 坏得多	原联邦德国、美国等
六级标准	1 不能察觉 2 刚察觉 3 明显但不妨碍 4 稍有妨碍 5 明显妨碍 6 极妨碍(不能用)	英国、欧洲国际广播联盟等	六级标准	6 优 5 良 4 中 3 稍次 2 次 1 极次	美国、欧洲国际广播联盟等	七级标准	+3 好得多 +2 好 +1 稍好 0 相同 -1 稍坏 -2 坏 -3 坏得多	欧洲国际广播联盟等

习题

4-1　举例说明日常生活中观察到的数字图像成像系统及其成像原理。

4-2　什么是线性系统？线性系统具有哪些重要性质？

4-3　试写出调谐信号的函数形式。

4-5　试写出传递函数的数学表达式，传递函数具有哪些重要性质？

4-6　若某系统为线性移不变系统，在传递函数的作用下，当输入信号频率为 5π 的调谐信号时，系统的输出信号的频率是多少？为什么？

4-7　试简述什么是抖动技术。

4-8　方阵具有哪些性质？

4-9　迄今为止为什么还用图像质量主观评价标准？它有何意义？

第 5 章 图像频域变换



在 MATLAB 中，一般用二元函数 $f(x,y)$ 作为图像的数学表示。$f(x,y)$ 表示在特定点 (x,y) 处的函数值，表示图像在该点相应的颜色强度或者灰度。所谓图像变换就是指把图像转换为另一种数学表示方法的操作。

在图像处理技术中，图像的正交点变换技术有着广泛的应用，是图像处理的重要工具。通过图像变换，改变图像的表示域及表示数据，可以给后续工作带来极大的方便。例如，傅里叶变换可使处理分析在频域中进行，使运算变得简单；而离散余弦变换（DCT）可使能量集中在少数数据上，从而实现数据压缩，便于图像传输和存储。

在 MATLAB 7.0 图像处理工具箱中，提供了几种常用的图像变换函数，它们是傅里叶变换（Fourier Transform），离散余弦变换（Discrete Cosine Transform）和 Radon 变换（Radon Transform）。另外，随着小波分析方法在图像处理中的应用不断发展成熟，MATLAB 小波分析工具箱也提供了很多小波变换函数，用于图像处理。鉴于篇幅限制，关于小波分析在图像处理中的应用在这只用一小节讲述，在第 9 章会展开来讲述。

5.1 傅里叶变换

5.1.1 傅里叶变换的基本概念

假设 $f(m,n)$ 是一个包含两个离散空间变量 m 和 n 的函数，则该函数的二维傅里叶变换的定义如下：

$$F(\omega_1,\omega_2)=\sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty}f(m,n)\mathrm{e}^{-\mathrm{j}\omega_1 m-\mathrm{j}\omega_2 m} \tag{5-1}$$

式中，ω_1 和 ω_2 为频域变量，其单位为弧度/采样单元。通常函数 $F(\omega_1,\omega_2)$ 称为函数 $f(m,n)$ 的频域表示。$F(\omega_1,\omega_2)$ 是复变函数，其变量 ω_1 和 ω_2 的周期均为 2π。因为这种周期性的存在，所以通常在图像显示时，这两个变量的取值范围为 $-\pi\leqslant\omega_1,\omega_2\leqslant\pi$。

傅里叶反变换的定义如下：

$$f(m,n)=\frac{1}{4\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}F(\omega_1,\omega_2)\mathrm{e}^{\mathrm{j}\omega_1 m}\mathrm{e}^{\mathrm{j}\omega_2 n}\mathrm{d}\omega_1\mathrm{d}\omega_2 \tag{5-2}$$

简单地说，该方程说明 $f(m,n)$ 可以表示为无限多项式不同频率的复指函数之和。而不同的频率点 (ω_1,ω_2) 所占的比例由幅度 $F(\omega_1,\omega_2)$ 决定。

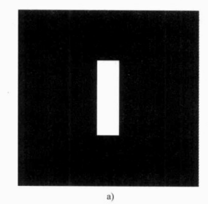
例如，考察下面的矩形函数 $f(m,n)$，该函数在一个矩形区域中的函数值为 1，在其他区域中的函数值都为 0，如图 5-1a 所示。

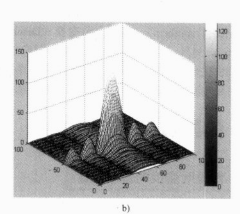
在 MATLAB 7.0 中，变量 m、n 和函数 $f(m,n)$ 均采用离散形式表示，所以要想真实地逼近连续函数，只能通过提高取样率来实现。因此，$f(m,n)$ 函数的傅里叶变换可由以下程序

段获得，傅里叶变换的幅值即$|F(\omega_1,\omega_2)|$，如图 5-1b 所示。其中 x 轴和 y 轴分别为水平分量和垂直分量。

```
clear
N=100
f=zeros(50,50);
f(15:35,23:28)=1;
figure,imshow(f,'notruesize');
F=fft2(f,N,N);
F2=fftshift(abs(F));
figure;
x=1:N;y=1:N;
mesh(x,y,F2(x,y));
colormap(gray);colorbar
```

执行程序代码后效果如图 5-1 所示。


a)



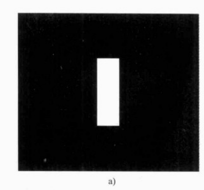
b)

图 5-1　矩形连续函数及其傅里叶变换的幅值

a) 矩形连续函数　b) 傅里叶变换的幅值

对于傅里叶变换的结果，通常还采用另一种方法进行显示，即将变换结果的函数值取对数，即$\log|F(\omega_1,\omega_2)|$（在 MATLAB 中，$\log|F(\omega_1,\omega_2)|$即为$\ln|F(\omega_1,\omega_2)|$，但习惯用$\log|F(\omega_1,\omega_2)|$）接近于 0 值部分的细节凸现出来，如图 5-2 所示。

```
clear
N=100
f=zeros(50,50);
f(15:35,23:28)=1;
figure,imshow(f,'notruesize');
F=fft2(f,N,N);
F2=log(abs(F));
figure;
x=1:N;y=1:N;
imshow(F2,[],'notruesize')
```


a)

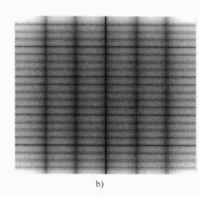
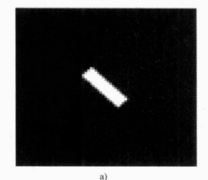

b)



图 5-2 傅里叶变换对数图形

a) 矩形连续函数 b) 傅里叶变换对数图形

对如图 5-1 所示的矩阵连续函数旋转一个角度，可得到如图 5-3a 所示的图像，然后对其进行傅里叶变换，得到如图 5-3b 所示的对数图。


a)

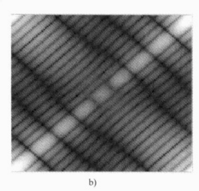

b)

图 5-3 矩形函数及其傅里叶变换的对数图

a) 旋转后的矩形函数 b) 傅里叶变换的对数图

```
clear
N=100
f=zeros(50,50);
f(15:35,23:28)=1;
J=imrotate(f,45,'bilinear')  %图像旋转 45°（5.2 节讲述）
figure,imshow(J,'notruesize')
F=fft2(J,N,N);
F2=log(abs(F));
figure;
x=1:N;y=1:N;
imshow(F2,[],'notruesize')
```

比较图 5-2 和图 5-3，可以发现二维傅里叶变换具有如下旋转性质：

如果在极坐标下表示二维函数图形，把空间域和空间频域的直角坐标均作坐标转换。

空间域 $x = r\cos\theta, y = r\sin\theta$ 空间频域 $u = \omega\cos\phi, y = \omega\sin\phi$

则有 $f(m,n)$ 在空间域极坐标系中表示为 $f(r,\theta)$，$F(u,v)$ 在空间频域极坐标系中表示为 $F(\omega,\phi)$。如果有

$$f(r,\theta) \Leftrightarrow F(\omega,\phi) \tag{5-3}$$

则有

$$f(r,\theta+\theta_0) \Leftrightarrow F(\omega,\phi+\phi_0) \tag{5-4}$$

即如果 $f(x,y)$ 在空间旋转一个角度 θ_0 后得到新的函数 $f(r,\theta+\theta_0)$，其对应的傅里叶变换 $F(\omega,\phi+\phi_0)$ 是 $f(x,y)$ 的傅里叶变换 $F(\omega,\phi)$ 在空间频域中旋转同样的角度 θ_0 得到的函数，反之亦然。

5.1.2 离散傅里叶变换

在用计算处理傅里叶变换通常采用离散傅里叶变换（Discrete Fourier Transform，DFT）。采用离散傅里叶变换主要有以下两个原因：

1）因为 DFT 的输入/输出均为离散值，非常适用于计算机的操作运算。

2）采用离散傅里叶变换，就可以用一种快速算法，即快速傅里叶变换（Fast Fourier Transform，FFT）。

FFT 的设计思想是将原函数分为奇数项和偶数项，通过不断将一个奇数项和一个偶数项相加（减），得到需要的结果。

也就是说 FFT 是将复杂的乘法运算变成两个数相加（减）的简单运算的重复，即通过计算两个单点的 DFT，来计算一个双点的 DFT；通过计算两个双点的 DFT，来计算四个点的 DFT……依此类推。

对于任何 $N = 2^m$ 的 DFT 计算，通过计算两个 $N/2$ 点的 DFT，来计算 N 个点的 DFT。

设离散函数 $f(m,n)$ 在有限区域（$0 \leqslant m \leqslant M-1$，$0 \leqslant n \leqslant N-1$）非零。则快速傅里叶变换的主要推导过程如下：

令

$$W_N^{pm} = \exp(-\mathrm{j}\frac{2\pi pm}{N}) \tag{5-5}$$

则有

$$\begin{aligned}
F(p) &= \frac{1}{N}\sum_{m=0}^{N-1} f(m)W_N^{pm} \\
&= \frac{1}{2}\left[\frac{2}{N}\sum_{n=0}^{N/2-1} f(2m)W_N^{2pm} + \frac{2}{N}\sum_{n=1}^{N/2-1} f(2m+1)W_N^{p(2m+1)}\right] \\
&= \frac{1}{2}\left[\frac{1}{M}\sum_{m=0}^{M-1} f(2m)W_N^{2pm} + \frac{1}{M}\sum_{m=1}^{M-1} f(2m+1)W_N^{2pm}W_N^{p}\right] \\
&= \frac{1}{2}[F_e(p) + W_N^p F_o(p)]
\end{aligned} \tag{5-6}$$

式（5-6）在计算时分成了奇数项和偶数项。

同理

$$F(p+M)=\frac{1}{2}\left[F_e(p+M)+W_N^{p+M}F_o(p+M)\right]$$
$$=\frac{1}{2}[F_e(p)+W_N^{p+M}F_o(p)] \tag{5-7}$$

又

$$W_N^{p+M}=W_N^pW_N^M=W_N^p\exp(-\mathrm{j}\frac{2\pi M}{N})=W_N^p\exp(-\mathrm{j}\pi)=-W_N^p \tag{5-8}$$

所以

$$F(p+M)=\frac{1}{2}\left[F_e(p)-W_N^pF_o(p)\right]$$

由式（5-7）和式（5-8）推导可得 FFT 的定义式为

$$F(p,q)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)\mathrm{e}^{-\mathrm{j}(2\pi/M)pm}\mathrm{e}^{-\mathrm{j}(2\pi/N)qn} \quad \begin{array}{l}p=0,1,\cdots,M-1\\ q=0,1,\cdots,N-1\end{array} \tag{5-9}$$

$$f(m,n)=\frac{1}{MN}\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}F(p,q)\mathrm{e}^{\mathrm{j}(2\pi/M)pm}\mathrm{e}^{\mathrm{j}(2\pi/N)qn} \quad \begin{array}{l}m=0,1,\cdots,M-1\\ n=0,1,\cdots,N-1\end{array} \tag{5-10}$$

可以认为，$F(p,q)$ 是 $f(m,n)$ 的 DFT 系数。

在 MATLAB 7.0 中，可分别用函数 fft()、fft2()和 fftn()来计算一维、二维和 n 维的 FFT，而其反变换依次为 ifft()、ifft2()和 ifftn()。

下面构建一个类似于图 5-1 所示的矩阵函数，程序代码如下：

```
f=zeros(30,30);
f(5:24,13:17)=1;
%显示效果如图 5-4 所示
figure, imshow(f,'notruesize');
F=fft2(f);
F2=log(abs(F));
%显示效果如图 5-5 所示
imshow(F2,[.1 5],'notruesize');
colormap(jet);colorbar
```

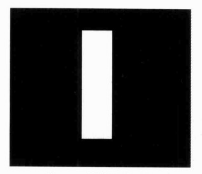


图 5-4 离散矩形函数

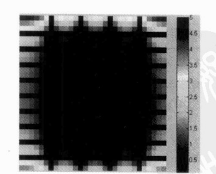


图 5-5 函数傅里叶变换幅值对数图

针对分辨率低和直流成分显示区域调整问题，MATLAB 7.0 分别提供了补零和 fftshift()函数来解决。

在计算离散函数的 DFT 时，可以通过对该函数进行补零来提高分辨率，补零和 DFT 计算可以在一个函数调用中实现：

```
F=fft2(f,M,N);
```

其中 M 和 N 是足够大的整数，用于指定覆盖矩阵的大小，超出部分将补零。如上面的例子，可以通过下面的程序段来提高分辨率。

```
F=fft2(f,256,256);
imshow(log(abs(F)),[.1,5]);
colormap(jet);colorbar;
```

结果如图 5-6 所示。

而 DC 系统的移动可用 fftshift()函数实现：

```
F=fft2(f,256,256);
F2=fftshift(F);
imshow(log(abs(F2)),[.1,5]);
colormap(jet);colorbar;
```

移动后的图像如图 5-7 所示。

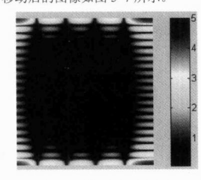


图 5-6　补零的傅里叶变换

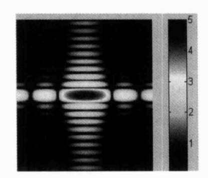


图 5-7　移位后的幅值

5.1.3 傅里叶变换的应用

傅里叶变换在图像处理中有着广泛的应用，下面以几个常用的例子进行简单的介绍。

1. 线性滤波器的频率响应

在进行滤波器设计时，通常需要分析所设计滤波器的频率响应特性，如通带、起始频率等，通过分析，看所设计的滤波器是否满足参数指标要求。

MATLAB 7.0 提供了 freqz2()函数，可以对线性滤波器的频率响应进行分析，这是因为线性滤波器冲击响应的傅里叶变换很好地反映了滤波器的频率响应特性。

以下程序代码示例给出了高斯滤波器的频率响应，并显示了滤波器的通带特性。

```
h=fspecial('gaussian');
freqz2(h)
```



运行程序效果如图 5-8 所示。

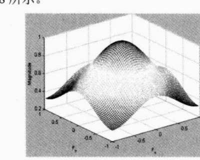
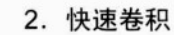


图 5-8　滤波器频率响应



2. 快速卷积

傅里叶变换的另一个重要特性是能够实现快速卷积。由线性系统理论可知，两个函数的卷积的傅里叶变换等于两个函数的傅里叶变换的乘积。该特性与快速傅里叶变换一起，可以快速计算函数的卷积。

假设 $\boldsymbol{A}$ 为 $M\times N$ 的矩阵，$\boldsymbol{B}$ 为 $P\times Q$ 的矩阵，则快速计算卷积的方法如下：

对 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 补零，使其大小都为 $(M+P-1)\times(N+Q-1)$。

利用 fft2()函数对 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 分别进行二维 FFT 变换。

将两个 FFT 结果相乘，利用 ifft2()函数对乘积进行傅里叶变换。

举例如下：

```
A=magic(3)
B=ones(3)
A(8,8)=0; %对矩阵 A 进行补零
B(8,8)=0; %对矩阵 B 进行补零
C=ifft2(fft2(A).*fft2(B));
C=C(1:5,1:5);
C=real(C)
```

程序执行的结果为

```
A =
     8     1     6
     3     5     7
     4     9     2
B =
     1     1     1
     1     1     1
     1     1     1
C =
    8.0000    9.0000   15.0000    7.0000    6.0000
   11.0000   17.0000   30.0000   19.0000   13.0000
   15.0000   30.0000   45.0000   30.0000   15.0000
    7.0000   21.0000   30.0000   23.0000    9.0000
    4.0000   13.0000   15.0000   11.0000    2.0000
```



其结果可与直接调用 MATLAB 7.0 中的卷积函数 C=conv2(A,B)相比较，直接用函数实现卷积结果如下：

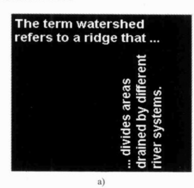
```
C=conv2(A,B);
C=C(1:5,1:5);
C=real(C)
C =
     8     9    15     7     6
    11    17    30    19    13
    15    30    45    30    15
     7    21    30    23     9
     4    13    15    11     2
```

经过比较，两者运算的结果相同。

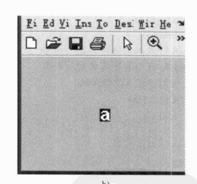
3. 图像特征定位

傅里叶变换可以用于与卷积密切相关的运算（Correlation）。数字图像处理中的相关运算通常用于匹配模板，还可对某些模板对应的特征进行定位。例如，假如用户希望在图像 text.tif 中定位字母“a”，如图 5-9a 所示，可以采用下面的方法定位。

将包含字母“a”的图像与图像 text.png 进行相关运算，也就是对包含字母“a”的图像和图像 text.png 进行傅里叶变换，然后利用快速卷积的方法，计算字母“a”和图像 text.png 的卷积，提取卷积运算的峰值，即得到在图像 text.png 中对应字母“a”的定位结果。执行程序如下（见图 5-9b）：



a)



b)

图 5-9 在图形中定位字母“a”

a) 原图 text. png b) 模板

```
bw=imread('text.png');
a=bw(32:45,88:98);
figure,imshow(bw);
figure,imshow(a);
```

所谓将模板“a”与图像 text.png 进行相关运算，就是先分别对其作快速傅里叶变换，然后利用快速卷积的方法，计算模板和图像 text.png 的卷积，如图 5-10 所示，并提取卷积运算结果的最大值，即图 5-11 所示的白色亮点，即得到图像 text.png 中字母“a”的定位结果。

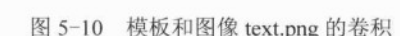


图 5-10　模板和图像 text.png 的卷积

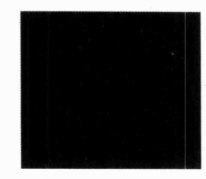


图 5-11　图像 text.png 中字母“a”的定位结果

```
bw=imread('text.png');
a=bw(32:45,88:98);  %从图像中提取字母“a”
C=real(ifft2(fft2(bw).*fft2(rot90(a,2),256,256)));
figure,imshow(C,[])
max(C(:))
thresh=60  %设定门限
figure,imshow(C>thresh)
ans =
    68
thresh =
    60
```

5.2 离散余弦变换

如果函数 $f(x)$ 为一个连续的实偶函数，即 $f(x)=f(-x)$，则此函数的傅里叶变换如下：

$$
\begin{aligned}
F(u) &= \int_{-\infty}^{+\infty} f(x)\mathrm{e}^{-\mathrm{j}2\pi ux}\mathrm{d}x \\
&= \int_{-\infty}^{+\infty} f(x)\cos(2\pi ux)\mathrm{d}x - \mathrm{j}\int_{-\infty}^{+\infty} f(x)\sin(2\pi ux)\mathrm{d}x \\
&= \int_{-\infty}^{+\infty} f(x)\cos(2\pi ux)\mathrm{d}x
\end{aligned}
\tag{5-11}
$$

因为虚部的被积项为奇函数，故傅里叶变换的虚数项为零，由于变换后的结果仅含有余弦项，故称为余弦变换。因此，余弦变换是傅里叶变换的特例。

5.2.1 一维离散余弦变换

离散余弦变换也是一种可分离变换，设 $\{f(x)\,|\,x=0,1,\cdots,N-1\}$ 为离散的信号序列，一维 DCT 变换对的定义如下：

$$C(u)=a(u)\sum_{x=0}^{N-1} f(x)\cos\frac{(2x+1)u\pi}{2N}\qquad (u=0,1,2,\cdots,N-1) \tag{5-12}$$

$$f(x)=\sum_{u=0}^{N-1} a(u)C(u)\cos\frac{(2x+1)u\pi}{2N}\qquad (x=0,1,2,\cdots,N-1) \tag{5-13}$$

式中，

$$a(u)=\begin{cases}\sqrt{1/N} & u=0\\ \sqrt{2/N} & \text{其他}\end{cases} \tag{5-14}$$

由一维离散余弦变换对的定义式可以看出，其正、反变换核均为

$$g(x,u)=h(x,u)=a(u)\cos\frac{(2x+1)u\pi}{2N}\quad (x,u=0,1,2,\cdots,N-1) \tag{5-15}$$

可见，一维 DCT 的逆变换核与正变换核是相同的。

5.2.2 二维离散余弦变换

考虑到两个变量，很容易将一维 DCT 的定义推广到二维 DCT。

设 $f(x,y)$ 为 $N\times N$ 的数字图像矩阵函数，则二维 DCT 变换对定义如下：

$$C(u,v)=a(u)a(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \tag{5-16}$$

式中，$u,v=0,1,2,\cdots,N-1$。

$$f(x,y)=a(u)a(v)\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}a(u)a(v)C(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \tag{5-17}$$

式中，$x,y=0,1,2,\cdots,N-1$。

由二维离散余弦变换对的定义式可以看出，其正、反变换核为

$$g(x,y,u,v)=h(x,y,u,v)=a(u)a(v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2u+1)v\pi}{2N} \tag{5-18}$$

式中，$a(u)$ 和 $a(v)$ 的定义同式（5-14）。

由此可知，DCT 的变换核具有可分离性，而且二维 DCT 的正反变换核是相同的。

与变换核为复指数的 DFT 相比，由于 DCT 的变换核是实数的余弦函数，因此 DCT 的计算速度要快，已广泛用于数字信号处理，如图像压缩编码、语音信号处理等方面。

5.2.3 快速离散余弦变换

关于 DCT 快速算法已经有多种方案，一种典型的算法就是利用 FFT。

一维 DCT 和 DFT 具有相似性，重写 DCT 如下：

$$C(0)=\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}f(x) \tag{5-19}$$

$$\begin{aligned}C(u)&=\sqrt{\frac{2}{N}}\mathrm{Re}\left\{\left[\exp\left(-\mathrm{j}\frac{u\pi}{N}\right)\right]\times\left[\sum_{x=0}^{2N-1}f_e(x)\exp\left(-\mathrm{j}\frac{2xu\pi}{2N}\right)\right]\right\}\\&=\sqrt{\frac{2}{N}}\mathrm{Re}\left\{\mathrm{e}^{-\mathrm{j}\frac{u\pi}{2N}}\left[\sum_{x=0}^{2N-1}f_e(x)\exp\left(-\mathrm{j}\frac{2xu\pi}{2N}\right)\right]\right\}\\&=\sqrt{\frac{2}{N}}\mathrm{Re}\left\{\omega^{\frac{u}{2}}\sum_{x=0}^{2N-1}f_e(x)\omega^{ux}\right\}\end{aligned} \tag{5-20}$$

式中，$\omega=\mathrm{e}^{-\mathrm{j}\frac{\pi}{N}}$ $\quad f_e(x)=\begin{cases}f(x) & x=0,1,2,\cdots,N-1\\ 0 & x=N,N+1,\cdots,2N-1\end{cases}$

对比 DFT 的定义可以看出，将序列拓展之后，DFT 变换的实部对应 DCT，而虚部对应着离散正弦变换，因此可以利用 FFT 实现 DCT。这种方法的缺点是将序列拓展了，增加了一些不必要的计算量，此外这种处理也容易造成误解。其实，DCT 是独立发展的，并不是源于 DFT 的。

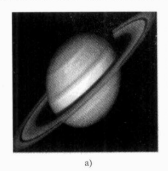
5.2.4 离散余弦应用

下面通过一段程序示例来说明 DCT 的一些性质，如图 5-12 所示。

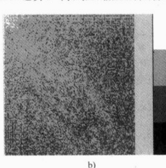
程序代码如下：

```
clear
RGB=imread('F:\image\baboon.bmp');
GRAY=rgb2gray(RGB);
figure,imshow(GRAY);
D=dct2(GRAY);
figure,imshow(log(abs(D)),[ ]);
colormap(gray(4));colorbar;
D(abs(D)<0.1)=0;
I=idct2(D)/255;
figure,imshow(I)
```

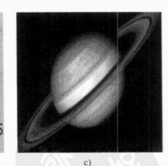
以上程序段对原始图像进行离散余弦变换，如图 5-12a 所示，变换后的结果如图 5-12b 所示。由结果可知，变换后 DCT 系数能量主要集中在左上角，其余大部分系数接近于零，这说明 DCT 具有适用于图像压缩的特性。将变换后的 DCT 系数进行门限操作，将小于一定值的系数归零，然后进行逆 DCT 运算，得到压缩后的图像，如图 5-12c 所示。



a)



b)



c)

图 5-12 余弦变换应用

a) 原始图像 b) 余弦变换结果 c) 压缩后的图像

比较变换前后的图像，可以发现视觉效果相差很小，可见压缩的效果比较理想。

在 JEPG 图像压缩算法中，其基本原理是：首先量化、编码。传输后，JPEG 接收端解码量化了的 DCT 系数，计算每一块的逆二维离散余弦变换，然后重组这些小块成为一幅图像。通常，经过变换后大部分 DCT 系数都近似为 0，因此，这些系数对重构的影响很小。

5.3 离散沃尔什—哈达玛变换（DWT—DHT）

前面介绍的傅里叶变换、离散余弦变换及后面将要介绍的 Radon 变换都是由指数函数，即以正弦、余弦三角函数为基本正交函数的级数展开而构成的。由于在解决实际问题时，频域法往往更容易处理，所以以上 3 种变换在信号处理、图像处理以及系统设计中得到很广泛的应用。

但是，傅里叶变换和离散余弦变换在快速算法中都要用到复数乘法，占用的时间仍然比较多，在某些应用领域中，则需要更为便利和有效的变换方法。为此，本节将介绍沃尔什（Walsh）变换，该变换就是比较有效的一种变换方法。

沃尔什变换是由两个数值（+1 和-1）作为基本函数的级数展开而成的，它满足正交特性。由于沃尔什函数是二值正交函数，两个数值与数字逻辑中的两个状态相对应，因此更适合在计算机技术、数字信号处理领域中应用。

沃尔什变换是在 1923 年由美国数学家沃尔什（Walsh）提出的，其原始论文中给出了沃尔什变换的递推公式，后来法国数学家哈达玛（Hadamard）等利用只包含+1 和-1 阵元的正交矩阵来表示沃尔什变换，所以沃尔什变换与哈达玛变换是等价的。与傅里叶变换相比，沃尔什变换的主要优点在于存储空间少和运算速度高，这一点对图像处理来说至关重要，特别是在大量数据需要进行实时处理时，沃尔什变换就更显示出它的优越性。

5.3.1 一维离散沃尔什变换

1. 沃尔什正交变换

设 $f(x)$ 表示 N 点的一维离散序列，则一维沃尔什正变换定义如下：

$$F(u)=\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}f(x)\ (-1)^{\sum_{i=0}^{n-1}b_i(x)b_{n-1-i}(u)} \qquad (5\text{-}21)$$

$$u=0,1,2,3,\cdots,N-1$$

其中，一维离散沃尔什正变换的变换核为

$$g(x,u)=\frac{1}{\sqrt{N}}\ (-1)^{\sum_{i=0}^{n-1}b_i(x)b_{n-1-i}(u)} \qquad (5\text{-}22)$$

式中，$u=0,1,2,3,\cdots,N-1; x=0,1,2,3,\cdots,N-1$，$N$ 是一维离散沃尔什正变换的阶数，$N=2^n$。$b_i(x)$ 是 x 的二进制数的第 i 位数值，取值为 0 或 1。如 i=6，由于 6 的二进制表示为 110，因此 $b_0(x)=0$，$b_1(x)=1$，$b_2(x)=1$。

2. 沃尔什逆变换

一维离散沃尔什逆变换定义如下：

$$f(x)=\frac{1}{\sqrt{N}}\sum_{u=0}^{N-1}F(u)\ (-1)^{\sum_{i=0}^{n-1}b_i(x)b_{n-1-i}(u)} \qquad (5\text{-}23)$$

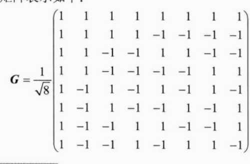
式中，$x=0,1,2,3,\cdots,N-1$。

其中，一维离散沃尔什逆变换的变换核为

$$h(x,u)=g(x,u) \qquad (5\text{-}24)$$

一维沃尔什正逆变换的变换核相同，沃尔什变换的变换核是一个对称阵列，其行和列是

正交的。沃尔什正、逆变换形式本质上相同，因此，计算沃尔什正变换的算法可直接用来求其逆变换。一维沃尔什正变换也具有快速算法，简称为 FWT，在形式上和 FFT 算法类似。当 N=8 时，其变换核用矩阵表示如下：



$$G=\frac{1}{\sqrt{8}}\begin{pmatrix}1&1&1&1&1&1&1&1\\1&1&1&1&-1&-1&-1&-1\\1&1&-1&-1&1&1&-1&-1\\1&1&-1&-1&-1&-1&1&1\\1&-1&1&-1&1&-1&1&-1\\1&-1&1&-1&-1&1&-1&1\\1&-1&-1&1&1&-1&-1&1\\1&-1&-1&1&-1&1&1&-1\end{pmatrix}$$



5.3.2 二维离散沃尔什变换

1. 二维沃尔什正变换

设 $f(x,y)$ 表示 $M\times N$ 的二维离散序列，则二维沃尔什正变换定义如下：

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)g(x,u,y,v) \tag{5-25}$$

$$u=0,1,2,3,\cdots,M-1;v=0,1,2,3,\cdots,N-1$$

式中，二维离散沃尔什正变换的变换核为

$$g(x,u,y,v)=\frac{1}{\sqrt{MN}}\sum_{u=0}^{N-1}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_{n-1-i}(u)]+\sum_{j=0}^{n-1}[b_j(x)b_{n-1-j}(u)]} \tag{5-26}$$

式中，$M=2^m;N=2^n$。

二维离散沃尔什正变换的变换核是可分离的，即

$$\begin{aligned}g(x,u,y,v)&=\frac{1}{\sqrt{MN}}\sum_{u=0}^{N-1}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_{n-1-i}(u)]+\sum_{j=0}^{n-1}[b_j(x)b_{n-1-j}(u)]} \qquad (5\text{-}27)\\&=\frac{1}{\sqrt{M}}\sum_{u=0}^{N-1}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_{n-1-i}(u)]}\frac{1}{\sqrt{N}}\sum_{u=0}^{N-1}(-1)^{\sum_{j=0}^{n-1}[b_i(x)b_{n-1-j}(u)]}\\&=g_1(x,u)g_2(y,v)\end{aligned}$$

根据沃尔什变换的定义形式可以得出，二维沃尔什变换具有可分离性，即一次二维沃尔什变换可以通过二次一维沃尔什变换来实现。

2. 二维沃尔什逆变换

二维沃尔什逆变换定义如下：

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)h(x,u,y,v) \tag{5-28}$$

$$x=0,1,2,3,\cdots,M-1;y=0,1,2,3,\cdots,N-1$$

二维离散沃尔什逆变换的变换核为

$$h(x,u,y,v)=g(x,u,y,v)$$

二维离散沃尔什逆变换的变换核为

$$g(x,u,y,v)=\frac{1}{\sqrt{MN}}\sum_{u=0}^{N-1}(-1)^{\sum_{i=0}^{m-1}[b_i(x)b_{n-1-i}(u)]+\sum_{j=0}^{m-1}[b_j(x)b_{n-1-j}(v)]} \tag{5-29}$$

式中，$M=2^m;N=2^n$。

同样，二维逆变换具有可分离性。二维沃尔什变换也可以表示为矩阵形式：

$$\begin{cases} F=\dfrac{1}{\sqrt{MN}}G_1 f G_2 \\ f=\dfrac{1}{\sqrt{MN}}G_1 F G_2 \end{cases} \tag{5-30}$$

式中，G_1 为 $M\times M$ 变换核方阵；G_2 为 $N\times N$ 变换核方阵。

沃尔什变换是将一个函数变换成取值为+1 或-1 的基本函数构成的级数，用它来逼近数字脉冲信号时要比 DFT 有利。因此，它在图像传输，通信技术和数据压缩中获得了广泛的应用。同时，沃尔什变换是实数，所以对工程应用问题，沃尔什变换的存储量比 DFT 少，而且运算速度非常快。

5.3.3 一维离散哈达玛变换

1. 一维哈达玛正变换

设 $f(x)$ 表示 N 点的一维离散序列，则一维哈达玛变换定义如下：

$$F(u)=\sum_{x=0}^{N-1}f(x)g(x,u)=\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}f(x)\ (-1)^{\sum_{i=0}^{n-1}b_i(x)b_i(u)} \tag{5-31}$$

$$u=0,1,2,3,\cdots,N-1$$

式中，$g(x,u)$ 是一维哈达玛变换的变换核，定义如下：

$$g(x,u)=\frac{1}{\sqrt{N}}\ (-1)^{\sum_{i=0}^{N-1}b_i(x)b_i(u)} \tag{5-32}$$

式中，$u=0,1,2,3,\cdots,N-1;x=0,1,2,3,\cdots,N-1$。$N$ 是哈达玛变换的阶数，$N=2^n$。$b_i(x)$ 是 x 的二进制数的第 i 位数值，取值为 0 或 1。

2. 一维哈达玛逆变换

若已知 N 点的一维离散序列 $F(u)$，则可以进行哈达玛逆变换，其定义如下：

$$f(x)=\frac{1}{\sqrt{N}}\sum_{u=0}^{N-1}F(u)\ (-1)^{\sum_{i=0}^{n-1}b_i(x)b_{n-1-i}(u)} \tag{5-33}$$

$$x=0,1,2,3,\cdots,N-1$$

与一维哈达玛正交变换相同，$h(x,u)$ 是一维哈达玛逆变换的变换核，逆变换的变换核与正交变换的变换核相等，即

$$h(x,u)=g(x,u)=\frac{1}{\sqrt{N}}(-1)^{\sum_{i=0}^{N-1}b_i(x)b_i(u)} \tag{5-34}$$

哈达玛变换的阶数具有规律性，即按照 $N=2^n$ 规律递升，高阶哈达玛矩阵可以通过低阶哈达玛矩阵的克罗尼科积运算求得，也就是说，哈达玛矩阵具有如下关系：

1）$\boldsymbol{H}_1=(1)$

2）$\boldsymbol{H}_2=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$

3） $\boldsymbol{H}_4=\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}$

4） $\boldsymbol{H}_N=\boldsymbol{H}_{2^n}=\boldsymbol{H}_2\boldsymbol{H}_{\frac{N}{2}}=\begin{pmatrix}H_{\frac{N}{2}}&H_{\frac{N}{2}}\\H_{\frac{N}{2}}&-H_{\frac{N}{2}}\end{pmatrix}$

采用上述规律求哈达玛变换矩阵要比直接用哈达玛变换的变换核求矩阵快得多，此结论提供了一种快速哈达玛变换，也可以称为 FHT。例如，根据哈达玛矩阵的运算规律，可以得出 8 阶哈达玛矩阵如下：

$$\boldsymbol{H}_8=\begin{pmatrix}H_4&H_4\\H_4&-H_4\end{pmatrix}=\frac{1}{\sqrt{8}}\begin{pmatrix}1&1&1&1&1&1&1&1\\1&-1&1&-1&1&-1&1&-1\\1&1&-1&-1&1&1&-1&-1\\1&-1&-1&1&1&-1&-1&1\\1&1&1&1&-1&-1&-1&-1\\1&-1&1&-1&-1&-1&1&1\\1&1&-1&-1&-1&-1&1&1\\1&-1&-1&1&-1&1&1&-1\end{pmatrix}$$

5.3.4 二维离散哈达玛变换

一维哈达玛变换可以很方便地推广到二维，其正变换定义如下：

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)g(x,u,y,v) \tag{5-35}$$

$$=\frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\ (-1)^{\sum_{i=0}^{m-1}b_i(x)b_{p_i(u)}+\sum_{i=0}^{n-1}b_i(y)b_{p_i(v)}}$$

逆变换为

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)h(x,u,y,v) \tag{5-36}$$

$$=\frac{1}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)\ (-1)^{\sum_{i=0}^{m-1}b_i(x)b_{p_i(u)}+\sum_{i=0}^{n-1}b_i(y)b_{p_i(v)}}$$

二维哈达玛正变换核为

$$g(x,u,y,v)=\frac{1}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}(-1)^{\sum_{i=0}^{m-1}b_i(x)b_{p_i(u)}+\sum_{i=0}^{n-1}b_i(y)b_{p_i(v)}} \tag{5-37}$$

式中，$x,u=0,1,2,3,\cdots,M-1;y,v=0,1,2,3,\cdots,N-1$。二维哈达玛逆变换核与正变换核相等，即

$$h(x,u,y,v)=g(x,u,y,v)$$

二维哈达玛变换核是可分离和对称的，因此一次二维哈达玛变换也可通过两次一维哈达玛变换来实现。

5.3.5 离散沃尔什—哈达玛变换的应用举例

下面是简单的离散沃尔什—哈达玛变换（DWT—DHT）的应用实例。

实例 1：有两个二维数字图像信号如下：

$$(1)\ \boldsymbol{f}_1=\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix}\qquad (2)\ \boldsymbol{f}_2=\begin{pmatrix}1&3&3&1\\1&3&3&1\\1&3&3&1\\1&3&3&1\end{pmatrix}$$

分别求 $\boldsymbol{f}_1$ 和 $\boldsymbol{f}_2$ 的二维沃尔什变换。

离散二维沃尔什变换的过程如下：

解：根据理论，对于（1）和（2），均有 $M=N=4$，因此，其二维沃尔什变换核为

$$\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}$$

由此可得

$$\boldsymbol{F}_1(\boldsymbol{u})=\frac{1}{16}\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}\times\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix}\times\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}$$

$$=\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$$

$$\boldsymbol{F}_2(\boldsymbol{u})=\frac{1}{16}\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}\times\begin{pmatrix}1&3&3&1\\1&3&3&1\\1&3&3&1\\1&3&3&1\end{pmatrix}\times\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}$$

$$=\begin{pmatrix}2&0&0&-1\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$$

实例 2：对简单数组进行沃尔什—哈达玛变换，程序如下：

```
clear
sq=[1 1 3 1
```



```
    2 1 2 2]
for k=1:4
    wht(:,k)=hadamard(2)*sq(:,k)/2
end
%%%%对每一列进行沃尔什—哈达玛变换，得到 wht
for j=1:2
    a=wht(j,:)'
    hadamard(4)
    wh(:,j)=hadamard(4)*wht(j,:)'/4
end
%%%%%%%%%的每一行进行沃尔什—哈达玛变换，得到 wh
wh=wh'  %重排
```

运行结果为

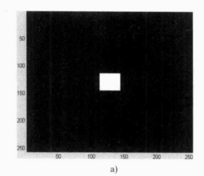
```
wht =
    1.5000    1.0000    2.5000    1.5000
   -0.5000         0    0.5000   -0.5000
wh =
    1.6250    0.3750   -0.3750   -0.1250
   -0.1250    0.1250   -0.1250   -0.3750
```

实例 3：对二维图像进行沃尔什—哈达玛变换，程序如下：

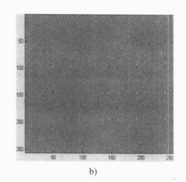
```
clear
I=zeros(2.^8);
I(2.^7-2.^4+1:2.^7+2.^4,2.^7-2.^4+1:2.^7+2.^4)=ones(2*2.^4);
figure;
colormap(gray(128)),imagesc(I);  %显示数据
[m,n]=size(I) %数据维数
for k=1:n
    wht(:,k)=hadamard(m)*I(:,k)/m;  %对每一列进行沃尔什—哈达玛变换
end
for j=1:m
    wh(:,j)=hadamard(n)*wht(j,:)'/n;  %对进行列的沃尔什—哈达玛变换后的
%系数进行沃尔什—哈达玛变换
end
wh=wh';
figure;
colormap(gray(128)),imagesc(wh);
```

程序运行结果如下（见图 5-13）：

```
m =
   256
n =
   256
```



a)



b)

图 5-13　对二维图像进行沃尔什—哈达玛变换

a) 生成原始图像　b) 沃尔什—哈达玛变换系数

5.4　K-L 变换

5.4.1　K-L 变换的定义

将一组离散信号变换为不相关数列的变换方法称为 Hotelling 变换。由于是由 H.Karhunen 和 M.Loeve 等人提出将连续信号变换为一组不相关数列的，所以也将 Hotelling 变换称为 K-L 变换。这种变换是建立在图像统计特性基础上的，其变换核矩阵由图像陈列协方差矩阵的特征值和特征向量决定，所以，K-L 变换也称为特征向量变换或主分量变换。

假定一幅 $N \times N$ 的数字图像通过某一信号通道传输了 M 次，由于受到各种因素的随机干扰，接收到的图像实际是一个受噪声干扰的数字图像集合。

$$\{f_1(x,y), f_2(x,y), \cdots, f_M(x,y)\}$$

写成 M 个 N^2 维向量 $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_i, \cdots, \boldsymbol{X}_M\}$ 的形式，其中，第 i 次获得了图像 $f_i(x,y)$ 所对应的 $\boldsymbol{X}_i$ 向量可采用行堆叠或列堆叠的方法构成，即

$$\boldsymbol{X}_i = \begin{pmatrix} f_i(0,0) \\ f_i(0,1) \\ \vdots \\ f_i(0,N-1) \\ f_i(1,0) \\ \vdots \\ f_i(1,N-1) \\ \vdots \\ f_i(N-1,0) \\ \vdots \\ f_i(N-1,N-1) \end{pmatrix} \tag{5-38}$$



$\boldsymbol{X}$ 向量的协方差矩阵定义为

$$C_f = E\{(X - m_f)(X - m_f)^{\mathrm{T}}\} \tag{5-39}$$

式中，E 为期望；T 为转置。

平均值向量 m_f 定义为

$$m_f = E\{X\} \tag{5-40}$$

对于 M 幅数字图像，平均值向量 m_f 和协方差矩阵 C_f 可由下述方法近似求得：

$$m_f = E\{X\} \approx \frac{1}{M}\sum_{i=1}^{M} X_i \tag{5-41}$$

$$C_f = E\{(X - m_f)(X - m_f)^{\mathrm{T}}\} \approx \frac{1}{M}\sum_{i=1}^{M}(X_i - m_f)(X_i - m_f)^{\mathrm{T}} \tag{5-42}$$

$$\approx \frac{1}{M}\sum_{i=1}^{M} X_i X_i^{\mathrm{T}} - m_f m_f^{\mathrm{T}}$$

由此可知，m_f 是 N^2 个元素的向量，C_f 是 $N^2 \times N^2$ 的方阵。

设 $\lambda_i (i = 1,2,\cdots,N^2)$ 是按递减顺序排列的协方差矩阵的特征值；$e_i = [e_{i1}, e_{i2}, \cdots, e_{iN^2}]^{\mathrm{T}}$ $(i = 1,2,\cdots,N^2)$ 是协方差矩阵的特征向量。

则定义 K-L 变换矩阵 A 为

$$A = \begin{pmatrix} e_{11} & e_{12} & \cdots & e_{1N^2} \\ e_{21} & e_{22} & \cdots & e_{2N^2} \\ \vdots & \vdots & & \vdots \\ e_{N^2 1} & e_{N^2 2} & \cdots & e_{N^2 N^2} \end{pmatrix} \tag{5-43}$$

从而可得 K-L 变换的表达式为

$$Y = A(X - m_f) \tag{5-44}$$

式中，$X - m_f$ 是原始图像向量 X 减去平均值向量 m_f，称为中心化图像向量。此式（5-44）表明，变换后的图像向量 Y 等于中心化图像向量 $X - m_f$ 与变换矩阵 A 的乘积。

5.4.2 K-L 变换的性质

1）变换后的图像向量 Y 的平均值向量 $m_Y = 0$，即为零向量。

证明：
$$m_Y = E\{Y\} = E\{A(X - m_f)\} = AE\{X\} - Am_f = 0 \tag{5-45}$$

2）Y 向量的协方差矩阵 $C_Y = E\{(Y - m_Y)(Y - m_Y)^{\mathrm{T}}\} = E\{YY^{\mathrm{T}}\}$

把式（5-44）代入，得

$$\begin{aligned} C_Y &= E\{(AX - Am_f)(AX - Am_f)^{\mathrm{T}}\} \\ &= E\{A(X - m_f)(X - m_f)A^{\mathrm{T}}\} \\ &= AE\{(X - m_f)(X - m_f)^{\mathrm{T}}\}A^{\mathrm{T}} \\ &= AC_f A^{\mathrm{T}} \end{aligned} \tag{5-46}$$

3）协方差矩阵 C_Y 是对角型矩阵，其对角线上的元素等于 C_f 的特征值 λ_i，

$i=1,2,\cdots,N^2$，即

$$C_Y=\begin{pmatrix}\lambda_1 & & & & & \\ & \lambda_2 & & & & \\ & & \ddots & & & \\ & & & \lambda_i & & \\ & & & & \ddots & \\ & & & & & \lambda_{N^2}\end{pmatrix} \tag{5-47}$$

C_Y 非对角线上的元素值为 0，说明变换后向量 Y 的像素是不相关的；而非对角线上的元素值不为 0，说明原始图像元素之间的相关性强。因此，K-L 变换的优点是去相关性好。

4）K-L 反变换。因为 C_f 是实对称矩阵，所以总可以找到一个标准正交的特征向量集合，使得 $A^{-1}=A^{\mathrm{T}}$，则可得到 K-L 反变换公式：

$$X=A^{-1}Y+m_f \tag{5-48}$$

总之，离散 K-L 变换的最大优点是去相关性好，可用于数据压缩和图像旋转。离散 K-L 变换应用的主要困难就是由协方差矩阵 C_f 求特征值和特征向量解方程的计算量问题；同时，K-L 变换是不可分离的，一般情况下，K-L 变换没有快速算法。

5.5 Radon 变换

5.5.1 Radon 变换原理

常用的 CT（X 射线计算机层析摄影仪）扫描，其原理是基于不同的物质有不同的 X 射线衰减系数，如果能确定人体衰减系数的分布，就能重建其断层或三维图像。但通过 X 射线透射时，只能测量到人体直线上的 X 射线衰减系数的平均值（是一个积分）。当直线变化时，此平均值（依赖于某参数）也随之变化，这样就不能很好地重建断层。但是，能否通过扫描测量的平均值求整个衰减系数的分布，从而有效地重建图像呢？Radon 变换为解决此类问题提供了很好的思路。

Radon()函数的作用是计算指定方向上图像的投影，对应的二元函数 $f(x,y)$，则是计算该函数在某一个方向上的线积分。例如，$f(x,y)$ 在垂直方向上的线积分即是 $f(x,y)$ 在 x 轴上的投影，而在水平方向上的线积分则是 $f(x,y)$ 在 y 轴上的投影，如图 5-14 所示。

而同样的投影可以沿任意角 θ 进行，一般来说，函数 $f(x,y)$ 的 Radon 变换就是沿着 y' 轴进行线积分。

Radon 变换的定义如下：

$$R_\theta(x')=\int_{-\infty}^{+\infty} f(x'\cos\theta-y'\sin\theta, x'\sin\theta+y'\cos\theta)\mathrm{d}y'$$
$$\begin{pmatrix}x'\\y'\end{pmatrix}=\begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} \tag{5-49}$$

Radon( )函数的几何关系示意图如图 5-15 所示。

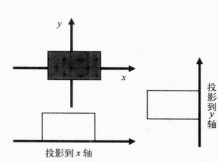




图 5-14 函数水平和垂直投影示意图

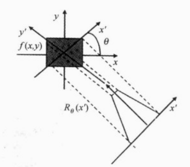
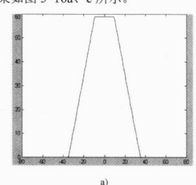


图 5-15 Radon 函数的几何关系示意图

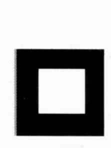
MATLAB 提供了函数[R,xp]=radon(I，theta)来计算图像 I 的 Radon 变换。其中 I 为待处理图像，theta 为投射角度。

其中，R 记录了图像 I 沿 theta 方向上的 Radon 变换值，xp 则对应于 x' 轴的坐标值。图像 I 的中心点位于 floor((size(I)+1)/2)，即 x' 轴上 $x'=0$。

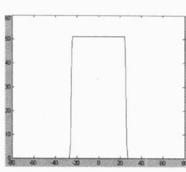
下面，利用 MATLAB 仿真程序计算图 5-16b 在 30° 和 90° 上的 Radon 变换。变换结果如图 5-16a、c 所示。



a)



b)



c)

图 5-16 30° 和 90° 方向上的 Radon 变换及原始图像

a) 30° 方向上的 Radon 变换 b) 产生原始图像 c) 90° 方向上的 Radon 变换

```
I=zeros(100,100);
I(25:75,25:75)=1;
figure,imshow(I);
[R,xp]=radon(I,[30 90]);
figure,plot(xp,R(:,1));
figure,plot(xp,R(:,2));
```

当然，也可以计算多个角度的 Radon 变换，并且表示为一幅图像。例如，可以沿 0°～180° 每隔 1° 作方形图的一组 Radon 变换，程序代码示例如下，变换结果如图 5-17 所示。

```
I=zeros(100,100);
I(25:75,25:75)=1;
theta=0:180;
[R,xp]=radon(I,theta);
```

```
imagesc(theta,xp,R);
xlabel('\theta(degrees)');
ylabel('X\prime');
set(gca,'XTick',0:20:180);
colormap(hot);
colorbar
```

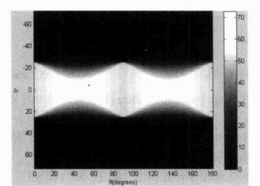


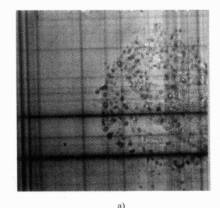
图 5-17　从 0°～180° 每隔 1° 进行 Radon 变换的结果

5.5.2 用 Radon 变换检测直线

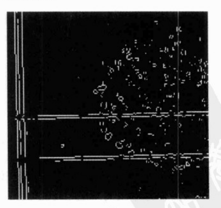
Radon 变换与计算机视觉中常用的 Hough 变换很相似。可以利用 Radon()函数来实现 Hough 变换，并进行直线检测。

以下程序示例将说明如何在 MATLAB 中通过程序仿真，利用 Radon 变换来实现直线的检测。

首先，对于已有的原始图像，如图 5-18a 所示，用 edge()函数计算该图像边缘的二值图像，边缘检测结果如图 5-18b 所示，得到候选的边缘线条。



a)



b)

图 5-18　用 Radon 变换进行边缘直线检测

a) 原始图像　b) 边缘检测结果



程序代码如下：

```
I=fitsread('solarspectra.fts');
I=mat2gray(I);
BW=edge(I);
figure,imshow(I);
figure,imshow(BW);
```

然后利用 Radon()函数，计算边缘图像的 Radon 变换，变换结果如图 5-19 所示。

程序代码如下：

```
I=fitsread('solarspectra.fts');
I=mat2gray(I);
BW=edge(I);
theta=0:179;
[R,xp]=radon(BW,theta);
figure,imagesc(theta,xp,R);colormap(hot);
xlabel('\theta(degrees)');ylabel('X\prime');
colorbar
```



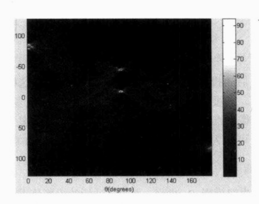


图 5-19　图 5-18 边缘的 Radon 变换结果

5.5.3 逆 Radon 变换及其应用

MATLAB 7.0 图像处理工具箱提供了 iradon()函数用于实现 Radon 变换，并经常用于投影成像值。这个变换能把 Radon 变换反变换回来，因此可以从投影数据重建原始图像。

给定图像 I 和一系列角度 theta，radon()函数可以用来计算图像的 Radon 变换，表达式为 R=Radon(R,theta)。计算逆 Radon 变换可以重建图像，其表达式为 IR=iradon(R,theta)。

这个重建图像的例子中，投影 R 是由原始图像 I 计算得到的。然而，在大多数的应用例子中，并没有得到当前投影的原始图像。例如，X 射线吸收断层摄影术，投影是通过测量 X 射线在不同角度通过物理切片时的衰减得到的。原始图像可以认为是切片的一个截面，图像的灰度代表切片的密度。投影通过特殊的硬件设备获得，而切片内部图像通过 iradon 函数重建。这样可以对活的生物体或不透明物体实现无损成像。

iradon()函数通过平行波束的投影来重建图像。在平行波束的几何关系中，每个投影通过

把以特定角度穿过图像的一系列线积分组合得到。如图 5-20 所示为平行波束几何在 X 射线吸收断层摄影术上的应用。注意，在平行波束的几何关系中，有相同数目的辐射器和接收器，辐射的衰减代表物体的密度、质量等的积分，这相当于 Radon 变换中的线积分。

图 5-20 中照射波束的关系是平行的，与 Radon 变换的几何关系相同。$f(x,y)$ 代表图像亮度，$R_\theta(x')$ 代表 θ 方向上的投影。

另外一种几何关系是扇形波束，即只有一个辐射器而有多个接收器，扇形波束投影可以转换成平行波束投影，利用逆 Radon 变换来重建图像。其几何关系如图 5-21 所示。

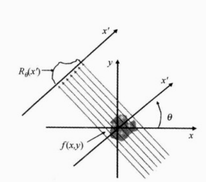


图 5-20　平行波束几何关系

图 5-21　扇形波束几何关系

iradon()函数利用滤波后的投影算法来计算逆 Radon 变换。这种算法利用 R 各列中的投影来构造图像 I 的近似值，若要获得更准确的图像，可以使用更多的投影值，投影数越多，重建的图像越接近原始图像。theta()矢量必须是固定增量的均匀矩阵，即角度每次的增量值 $\Delta\theta$ 为常数。若 $\Delta\theta$ 已知，可作为参数取代 theta 值，传入 iradon()函数。例如：

```
IR=iradon(R, Dtheta);
```

滤波 Radon 变换是首先对投影 R 滤波，再用滤波后的投影值重建图像。有些情况下，投影值含有噪声。Iradon()函数中可以采用多种窗函数，下面的例子采用了 hamming 窗函数来滤波。

```
IR=iradon(R ,theta,'hamming');
```

iradon()函数也允许指定归一化频率 D，高于 D 的滤波器响应为 0，整个滤波器压缩在 [0，D]内。这在投影时可以有效抑制高频噪声的干扰，而减少对图像重建的影响。下面调用 iradon()函数，设定归一化频率为 0. 85。

```
IR=iradon(R, theta ,0. 85);
```

接下来介绍如何利用 Radon()函数和 iradon()函数构造一个简单图像的投影并重建图像。

测试图像是 Shepp-Logan 的大脑幻影图，利用 MATLAB 7.0 图像处理工具箱的 phantom()函数可以产生数据。Shepp-Logan 的大脑幻影图反映了真实世界中人类大脑的很多性质。图像中外部的椭圆形是头骨，内部的椭圆是人脑的内部特征或者是肿瘤。

首先，产生如图 5-22 所示的 256 个灰度等级的大脑幻影图，相应的程序命令如下：

```
P=phantom(256);
figure,imshow(P);
```

然后计算 3 个不同分辨率的大脑幻影图的 Radon 变换，R1 有 18 条投影光束，R2 有 36 条投影光束，R3 有 90 条投影光束：

```
theta1=0:10:170;
[R1,xp]=radon(P,theta1);
theta2=0:5:175;
[R2,xp]=radon(P,theta2);
theta3=0:2:178;
[R3,xp]=radon(P,theta3);
```

显示 Shepp-Logan 大脑幻影图的有 90 条投影光束的 Radon 变换图形，即 R3 的图形。结果如图 5-23 所示。

```
figure,imagesc(theta3,xp,R3);
colormap(hot);colorbar;
xlabel('\theta');ylabel('x\prime')
```

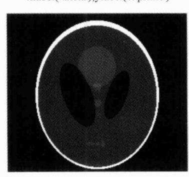


图 5-22　大脑的幻影图

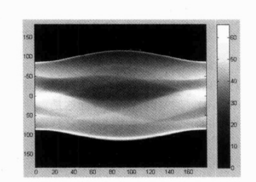
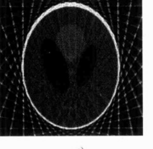
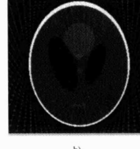


图 5-23　大脑幻影图的 90 条投影光束的 Radon 变换

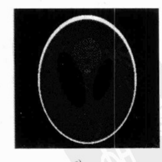
利用 R1、R2 和 R3 分别进行 Shepp-Logan 大脑幻影图的重建，结果如图 5-24 所示。



a)



b)



c)

图 5-24　重建图像

a) 用 R1 重建图像　b) 用 R2 重建图像　c) 用 R3 重建图像

```
I1=iradon(R1,10);
I2=iradon(R2,5);
I3=iradon(R3,2);
figure,imshow(I1);
figure,imshow(I2);
figure,imshow(I3)
```

5.6 小波变换

小波变换编码是近年来随着小波变换理论的研究而提出的一种具有很好发展前景的编码方法。作为一种多分辨率分析方法，由于小波变换具有很好的时-频和空-频局部特性，特别适合按照人类视觉系统的特性设计图像压缩编码方案，也非常有利于图像的分层传输。实验证明，图像的小波变换编码，在压缩比和编码质量方面优于传统的 DCT 变换编码。

5.6.1 传统变换方法的局限性

传统变换方法在信号分析中存在着许多不足之处，这是小波变换方法的研究越来越受到人们重视的一个重要原因。和小波变换相比，传统变换方法的局限性主要表现在以下两个方面。

1．对瞬态和局部信号分量的分析

以最具有代表性的傅里叶变换为例，其正交基函数是正弦信号，又叫正弦波，即这是一种波（Wave）。这是因为，一方面这是一种波动，即等幅振荡；另一方面它在两个方向上都是无限延伸的，就像海洋中的波浪一样。

瞬态信号只在很短的间隔上是非零的，而图像中的许多重要特征，如边缘等，在空间位置上都是高度局部性的。这些瞬态或局部信号分量和傅里叶变换的任何基函数都毫无相似之处，因而不能由其变换系数紧密地表示，这就使傅里叶变换和其他传统的基于波的变换在分析和压缩含有信号分量时性能不佳。

2．时-频和空-频局部化

在图像分析时，有时需要将信号在时域和频域中的特性或信号在空域和频域中的特性结合起来进行分析。例如，要了解图像的哪一部分含有较多的高频分量，或者某一段频率分量的分布情况等，这都是传统变换方法所无法解决的。

虽然傅里叶变换能够将任何解析函数甚至很窄的瞬态信号表示为正弦波之和，然而这要靠若干正弦波的复杂组合才能形成一个在大部分区间上为零的函数，这虽然是使变换成为可逆的有效方法，但却使函数的频谱与函数本身看起来截然不同。

在注意到傅里叶变换的弱点后，Gabor 于 1946 年提出了信号的时-频局部化的分析方法，就是人们通常说的 Gabor 变换，也称为加窗的傅里叶变换。

信号 $f(t)$ 的 Gabor 变换定义如下：

$$Wf(\omega,\sigma)=\int_R g(t-\sigma)f(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \qquad (5-50)$$

式中，函数 $g(\tau)=\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{r^2}{2}}$ 被称为窗函数。变换被限定在窗口中进行，比起在无限大空间的傅里叶变换有所进步，但是仍有局限性。因为无论 $g(\tau)$ 是什么样的窗函数，时窗 $g(t)$ 的宽度与频窗 $\hat{g}(\omega)$ 的宽度之积不小于 $1/\pi$，所以当确定某一个窗函数后，若其频宽对应于某一个频段，其时宽则不能太窄。为了提高局部的可观察性，则需要加大窗口，这样导致计算量大增，以致无法具体实现。

为了克服上述缺点，数学家们探索采用有限区间上的基函数进行变换。这些基函数不仅频率是可变的，而且位置也是可变的，这就是小波——有限区间上的波，如图 5-25 所示给出了小波曲线。小波之所以小，是因为它有衰减性，即是局部非零的；而称为波，则是因为它有波动性，即其取值呈正负相间的振荡形式。由于小波在频率和时间或空间位置上都是可变的，所以具有很好的时-频或空-频局部特性。

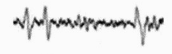


图 5-25 小波曲线

5.6.2 小波变换的基本知识

1. 连续小波变换（CWT）

所谓小波（Wavelet），即存在于一个较小区域的波。小波函数的数学定义是：设 $\psi(t)$ 为平方可积函数，即 $\psi(t)\in L^2(R)$，若其傅里叶变换 $\psi(\omega)$ 满足条件：

$$\int_R \frac{|\psi(\omega)^2|}{\omega}\mathrm{d}\omega<\infty \tag{5-51}$$

则称 $\psi(t)$ 为一个基本小波或波母函数，并称式（5-48）是小波函数的可允许条件。

根据小波函数的定义，小波函数一般在时域具有紧支集或近似紧支集，即函数的非零值定义域具有有限的范围，这即所谓“小”的特点。另一方面，根据可允许性条件可知 $\psi(\omega)|_{\omega=0}=0$，即直流分量为零，因此小波具有正负交替的波动性。

将小波母函数 $\psi(t)$ 进行伸缩和平移，设其伸缩因子（亦称尺度因子）为 a，平移因子为 τ，并记平移伸缩后的函数为 $\psi_{a,\tau}(t)$，则：

$$\psi_{a,\tau}(t)=a^{-1/2}\psi\left(\frac{t-\tau}{a}\right)\qquad (a>0,\tau\in R) \tag{5-52}$$

并称 $\psi_{a,\tau}(t)$ 为参数 a 和 τ 的小波基函数。由于 a 和 τ 均取连续变化的值，因此又称为连续小波基函数，它们是由同一母函数 $\psi(t)$ 经伸缩和平移后得到的一组函数。

将 $L^2(R)$ 空间的任意函数 $f(t)$ 在小波基函数下展开，称其为函数 $f(t)$ 的连续小波变换（CWT），变换式为

$$WT_f(a,\tau)=\frac{1}{\sqrt{a}}\int_R f(t)\overline{\psi\left(\frac{t-\tau}{a}\right)}\mathrm{d}t \tag{5-53}$$

式中，$\overline{\psi\left(\frac{t-\tau}{a}\right)\mathrm{d}t}$ 为小波基函数的共轭函数。

CWT 的变换结果是许多小波系数 $WT_f(a,\tau)$，这些系数是缩放因子和平移的函数。小波变换是通过缩放母小波的宽度来获得信号的频率特征，通过平移母小波来获得信号的时间信息。对母小波的缩放和平移操作是为了计算小波系数，这些小波系数反映了小波和局部信号之间的相关程度。

其小波母函数 $\psi(t)$ 缩放和平移的操作含义如下。

缩放就是压缩或伸展基本小波。小波的缩放因子与信号频率之间的关系是：缩放因子越小，小波越窄，度量的是信号的细节变化，表示信号频率越高；缩放因子越大，小波越宽，度量的是信号的粗糙程度，表示信号频率越低。小波的缩放操作如图 5-26 所示。

平移就是小波的延迟或超前，如图 5-27 所示。

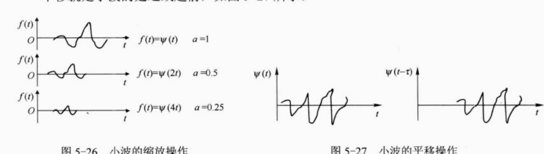


图 5-26　小波的缩放操作

图 5-27　小波的平移操作

CWT 计算主要有如下 5 个步骤：

1）取一个小波，将其与原始信号的开始一节进行比较。

2）计算数值 WT_f 。WT_f 表示小波与所取一节信号的相似程度。计算结果取决于所选小波的形状。

3）移动小波，重复第 1）步和第 2）步，直到覆盖整个信号。

4）伸展小波，重复 1）～3）步。

5）对于所有缩放，重复 1）～4）步。

在具体应用中，需要根据原函数 $f(t)$ 的特点来选择小波基函数 $\psi(t)$，使得小波变换能更好地反映函数 $f(t)$ 的特征，下面是一些小波基函数的例子。

1）哈尔（Haar）小波，如图 5-28 所示。其表达式为

$$\psi(t)=\begin{cases}1 & 0\leqslant t<1/2\\ -1 & 1/2\leqslant t<1\\ 0 & 其他\end{cases} \tag{5-54}$$

2）墨西哥帽小波，如图 5-29 所示。其表达式为

$$\psi(t)=\frac{2}{\sqrt{3\sqrt{\pi}}}(1-t^2)\mathrm{e}^{-\frac{t^2}{2}} \tag{5-55}$$



图 5-28 Haar 小波

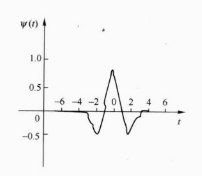


图 5-29 墨西哥帽小波

2. 离散小波变换（DWT）

在计算机应用中，连续小波应该离散化，这里的离散化是针对连续尺度参数 a 和连续平移参数 τ 的，而不是针对时间变量 t 的。为了使小波变换具有可变化的时间和频率分辨率，常常需要改变尺度参数 a 和平移参数 τ 的大小，即采用动态采样网格，以使小波变换具有“变焦距”的功能。小波分解的意义在于能够在不同尺度上对信号进行分析，而且对不同尺度的选择可以根据不同的目的来确定。在这种意义下，小波变换被称为数学显微镜。这就使分析十分有效，并且也是相当精确的，因此就得到所谓的离散小波变换。

实际上，人们是在一定尺度上认识信号的。人的感官和物理仪器都有一定的分辨率，对低于一定尺度信号的细节是无法认识的，因此对低于一定尺度信号的研究也是没有意义的。因此，应该将信号分解为对应不同尺度的近似分量和细节分量。信号的近似分量是大的缩放因子计算的系数，一般为信号的低频分量，包含着信号的主要特征；细节分量是小的缩放因子计算的系数，一般为信号的高频分量，给出信号的细节或差别。对信号的小波分解可以等效于信号通过了一个滤波器组，其中一个滤波器为低通滤波器，另一个为高通滤波器，分别得到信号的近似值和细节值，如图 5-30 所示。

由图 5-30 可以看出，离散小波变换可以表示成由低通滤波器和高通滤波器组成的一棵树。原始信号经过一对互补的滤波器组进行的分解称为一级分解，信号的分解过程也可以不断进行下去，也就是说可以进行多级分解。如果对信号的高频分量不再分解，而对低频分量进行连续分解，就可以得到信号不同分辨率下的低频分量，这也称为信号的多分辨率分析。如图 5-31 所示就是这样一个小波分解树。图中 S 表示原始信号，A 表示近似分量，D 表示细节分量，下标表示分解的层数。由于分析过程是重复迭代的，从理论上讲可以无限地连续分解下去，但事实上，分解可以进行到细节只包含单个样本为止。实际中，分解的级数取决于要分析的信号数据特征及用户的具体需要。

对于一个信号采用如图 5-30 所示的方法，理论上将产生两倍于原始数据的数据量。为此，根据奈奎斯特采样定理，采用下采样的方法来减少数据量，即在每个通道内（低通和高通通道），每两个样本数据取一个，通过计算得到离散小波变换系数，从而得到原始信号的近似分量与细节分量。

3. 逆离散小波变换（小波重构）

将信号的小波分解的分量进行处理后，一般还要根据需要把信号恢复出来，也就是利用信号的小波分解的系数还原出原始信号，这一过程称为逆离散小波变换，也常常称为小波重

构。小波分解包括滤波与下采样，小波重构过程则包括上采样与滤波。上采样的过程是在两个样本之间插入0。由图5-31可知，小波重构过程为$A_3+D_3=A_2$；$A_2+D_2=A_1$；$A_1+D_1=S$。

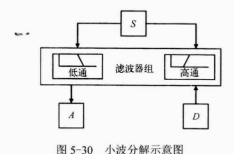


图5-30　小波分解示意图

图5-31　小波分解树

4．小波包分析

在小波分解中，一个信号可以不断分解为近似分量和细节分量，近似分量可以继续分解，但是细节分量不能分解，为此，人们又提出了对分量的小波包分解。使用小波包分解，不但可以不断分解近似分量，也可以继续分解细节分量，从而使整个分解构成一种二叉树结构，如图5-32所示。

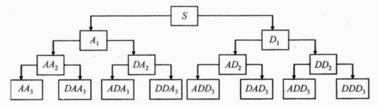


图5-32　小波包分解示意图

5．二维离散小波变换

二维离散小波变换是一维离散小波变换的推广，其实质是将二维信号在不同尺度上的分解。得到原始信号的近似分量和细节分量。由于信号是二维的，因此分解也是二维的。分解结果是：近似分量A、水平细节分量H、垂直细节分量V和对角细节分量D。同样也可以利用二维小波分解的结果在不同尺度上重构信号。

5.6.3　小波变换在图像处理方面的应用及实现

1．小波变换在图像处理的应用

小波变换是一种复杂的数学变换，可以在时域和频域上对原始信号进行多分辨率分解，小波变换的应用是与小波变换的理论研究紧密结合在一起的。小波变换在图像处理方面的应用十分广泛，可用于图像压缩、分类识别、去除噪声等；在医学成像方面，它用于减少B超、CT、核磁共振成像的时间，提高分辨率等。

小波变换用于信号与图像压缩，是小波变换应用的一个重要方面。它的特点是压缩比高，压缩速度快，压缩后能保持信号与图像的特征不变，且在传递中可以抗干扰。基于小波变换的压缩方法很多，比较成功的有小波包最优基方法、小波域纹理模型方法、小波变换零树压缩、小波变换向量压缩等。下面举一个二级小波分解的例子来说明基于小波变换的图像

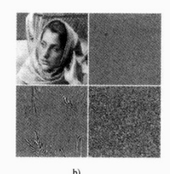


编码能够很好地实现图像分辨率和图像质量的多级伸缩性。

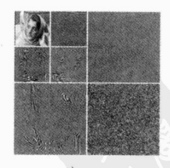
如图 5-33a 所示是一个分辨率为 256 像素×256 像素的灰度图像，图像的灰度级为 256，对这个二维原始图像进行小波变换，实际上就是把原始图像的像素值矩阵变换成另一个有利于压缩编码的系数矩阵。该系数矩阵所对应的图像如图 5-33b 所示，可以看出，经过一级小波变换后，原始图像被分解成几个子图像，每个子图像包含了原始图像中不同的频率成分。左上角子图包含了图像的低频分量，即图像的主要特征，低频分量可再次分解；右上角子图像包含了图像的垂直分量，即包含了较多的垂直边缘信息；左下角子图像包含了图像的水平分量，即包含了较多的水平边缘信息；右下角子图像包含了图像的对角分量，即同时包含了垂直和水平边缘信息。从图 5-33b 中可以看出，经过小波变换，原始图像的全部信息被重新分配到了 4 个子图中。左上角子图像包含了原始图像的低频信息，但失去了一部分边沿细节信息，这些失去的细节信息被分配到其他 3 个子图像中。由于失去了部分细节信息，所以左上角子图像比原始图像模糊了一些，不仅如此，其长宽尺寸也降低到原来的一半，即分辨率降低到原来的 1/4。一种最容易理解的图像压缩方法就是，丢弃 3 个细节子图像，只保留并编码低频子图像。但实际上，并不是通过这么简单的处理来进行图像压缩，3 个细节子图像不会被丢掉，而是与低频子图像一起编入码流，这样才可能在解码时恢复出完整的原始图像。当然，如果用户只需要一个小尺寸的图像，那就只需从码流中解码出低频子图像即可。低频子图像可以进一步分解，经过二级分解后，系数矩阵所对应的图像如图 5-33c 所示。图 5-33c 中，低频子图像的尺寸降到原始图像的 1/16，可见每一级分解都是对空间分辨率和频率分量的进一步细分。从此例可以看出，小波变换为在一个码流中实现图像多级分辨率提供了基础。前面提到，为了能在解码端恢复出完整的原始图像，所有的细节子图像都一起编入了码流，不扔掉这些细节，那图像的数据量又怎能被压缩呢？对图像进行了小波变换，并不代表图像的数据量就被压缩了，因为变换后，系数的总量并未减少，那么变换的意义何在呢？其意义在于使图像的能量分布（频域内的系数分布）发生改变，从而利于压缩编码。要真正地压缩数据量，还要对变换后的系数进行量化、扫描和熵编码，这样就可以达到减少图像数据量的目的。



a)



b)



c)

图 5-33　二级小波变换示例

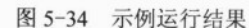
a) 原始灰度图像　b) 一级小波变换后的图像　c) 二级小波变换后的图像

2. 小波变换的 MATLAB 实现

利用二维小波变换对图像编码，如图 5-34 所示为运行结果，程序代码如下：

```
clear;              %清除 MATLAB 工作环境中现有的变量
load wbarb;         %装入图像
subplot(2,2,1);image(X);colormap(map);
title('原始图像');
disp('原始图像 X 的大小:');
whos('X');
%对图像用 bior3.7 小波进行二层小波分解
[c,s]=wavedec2(X,2,'bior3.7');
%提取小波分解结构中第一层的低频系数和高频系数
ca1=appcoef2(c,s,'bior3.7',1);
ch1=detcoef2('h',c,s,1);
cv1=detcoef2('v',c,s,1);
cd1=detcoef2('d',c,s,1);
%分别对各频率成分进行重构
a1=wrcoef2('a',c,s,'bior3.7',1);
h1=wrcoef2('h',c,s,'bior3.7',1);
v1=wrcoef2('v',c,s,'bior3.7',1);
d1=wrcoef2('d',c,s,'bior3.7',1);
c1=[a1,h1;v1,d1];
%显示分解后各频率分量的信息
subplot(2,2,2);image(c1);
axis square;
title('分解后低频和高频信息');
%下面进行图像压缩处理
%保留小波分解第一层低频信息，进行图像的压缩
%第一层的低频信息即 ca1，显示第一层的低频信息
%首先对第一层信息进行量化编码
ca1=appcoef2(c,s,'bior3.7',1);
ca1=wcodemat(ca1,440,'mat',0);
%改变图像的高度
ca1=0.5*ca1;
subplot(2,2,3);image(ca1);colormap(map);
axis square;
title('第一次压缩图像');
disp('第一次压缩图像的大小:');
whos('ca1');
% 保留小波分析分解第二层低频信息，进行图像的压缩，此时压缩比更大
% 第二层的低频信息 ca2，显示第二层的低频信息
ca2=appcoef2(c,s,'bior3.7',2);
ca2=wcodemat(ca2,440,'mat',0);
ca2=0.25*ca2;
subplot(2,2,4);image(ca2);colormap(map);
axis square;
title('第二次压缩图像');
disp('第二次压缩图像的大小:');
whos('ca2');
```




图 5-34 示例运行结果

原始图像 X 的大小:

```
Name    Size          Bytes   Class
X       256×256       524288  double array
Grand total is 65536 elements using 524288 bytes
```

第一次压缩图像的大小:

```
Name    Size          Bytes   Class
ca1     135×135       145800  double array
Grand total is 18225 elements using 145800 bytes
```

第二次压缩图像的大小:

```
Name    Size          Bytes   Class
ca2     75×75         45000   double array
Grand total is 5625 elements using 45000 bytes
```

5.7 扇形光束投影

类似于 Radon 变换，扇形光束投影（Fan-Beam Projection）也是通过求沿一组特定方向的直线积分来获得图像的映射，如对于一幅图像，就是求一组扇形直线的线积分，而这组扇形直线是由一个点源发散形成一个扇形而得名。为了获得图像的有效投影，可以改变点源的方向角，得到一组映射，如图 5-35 所示。

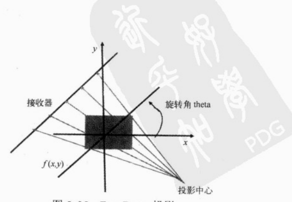


图 5-35 Fan-Beam 投影

MATLAB 提供了 fanbeam()函数来实现扇形光束投影的功能，下面分别介绍如何使用该函数获得映射数据以及从映射数据重建原图像，最后

给出一个例子来详细说明。

5.7.1 投影变换的基本概念

调用 fanbeam()函数实现图像映射，必须给定一些参数，如图像，扇形光束的原点以及旋转中心（即图像中心像素），投影线的数量由 fanbeam()函数根据图像的大小以及以上给定的每隔一个弧度（Arc）分配一条投影线，如图 5-36 所示。

如果将几何关系设定为“线形”，则其几何示意图如图 5-37 所示。

在获得图像的投影数据后，可以调用 ifanbeam( )函数来重建图像。

```
I=ifanbeam(P, D);
```

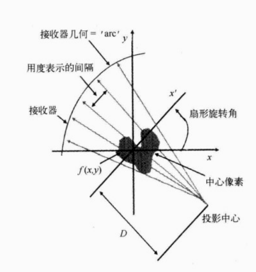


图 5-36 弧度光束投影原理图

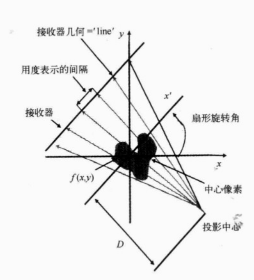


图 5-37 线性光束投影原理图

5.7.2 投影变换函数的应用

下面以一个实例来说明映射和重建的过程。

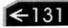
```
%产生测试图像并显示
P=phantom(256);
figure,imshow(P);
%计算投影数据
%设定几何关系为“线形”，分别给定光束数目为 18，36 和 90
[R1,xp]=radon(P,theta1);
num_angles_R1=size(R1,2) % num_angles_R1=18
theta2=0:5:175;
[R2,xp]=radon(P,theta2);
num_angles_R1=size(R2,2) % num_angles_R1=36
theta3=0:2:178;
```

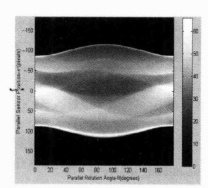
```
[R3,xp]=radon(P,theta3);
num_angles_R1=size(R3,2) % num_angles_R1=90
%设定几何关系为“弧度”，分别给定光束密度为 2 弧度、1 弧度以及 0.25 弧度
D=250;
dsensor1=2;
F1=fanbeam(P,D,'FanSensorSpacing',dsensor1);
dsensor2=1;
F2=fanbeam(P,D,'FanSensorSpacing',dsensor2);
dsensor3=0.25;
[F3,sensor_pos3,fan_rot_angles3]=fanbeam(P,D,'FanSensorSpacing',dsensor3);
%显示投影数据
%几何关系为“线形”
figure,imagesc(theta3,xp,R3)
colormap(hot)
colorbar
xlabel('Parallel Rotation Angle-\theta(degrees)');
ylabel('Parallel Sensor Position-x\prime(pixels)');
%几何关系为弧度
figure,imagesc(fan_rot_angles3,sensor_pos3,F3)
colormap(hot);colorbar
xlabel('Fan Rotation Angle(degrees)');
ylabel('Fan Sensor Position(degrees)');
%重建图像
%几何关系为“线形”
output_size=max(size(P));
dtheta1=theta1(2)-theta1(1);
I1=Iradon(R1,dtheta1,output_size);
figure,imshow(I1)
dtheta2=theta2(2)-theta2(1);
I2=Iradon(R2,dtheta2,output_size);
figure,imshow(I2)
dtheta3=theta3(2)-theta3(1);
I3=Iradon(R3,dtheta3,output_size);
figure,imshow(I3)
%几何关系为“弧度”
Ifan1=Ifanbeam(F1,D,'FanSensorSpacing',dsensor1,'OutputSize',output_size);
figure,imshow(Ifan1)
Ifan2=Ifanbeam(F2,D,'FanSensorSpacing',dsensor2,'OutputSize',outp
ut_size);
figure,imshow(Ifan2)
Ifan3=Ifanbeam(F3,D,'FanSensorSpacing',dsensor3,'OutputSize',outp
ut_size);
figure,imshow(Ifan3)
```



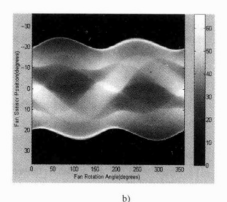
图 5-38 产生的原始图像

图 5-38 为由函数 phantom()产生的原始图像，图 5-39 为生成的投影数据，图 5-40 给出了不同投影密度重建的图像。





a)



b)

图 5-39　投影数据

a) 几何关系为“线形”　b) 几何关系为“弧形”

比较图 5-40 的 a～c 三幅图像，可以看到 I1、I2 和 I3 都不同程度存在虚假点，这三幅图像中 I3 的效果最好。比较图 5-40d～f 三幅图像，可以看到 Ifan1、Ifan2 和 Ifan3 都不同程度地存在模糊现象，Ifan3 的重建效果最好，而 Ifan1 模糊比较严重。图像重建中存在虚假点和模糊现象都与用于图像重建的投影光束的数目有关，投影光束越多，图像重建时就能更好地体现图像的细节，而且避免出现虚假点。显然，用 R1 和 F1 重建图像的投影光束太少，得到的重建结果明显比原始图像差很多。因此，为了获得高质量的重建结果，需要提高重建图像的投影光束。

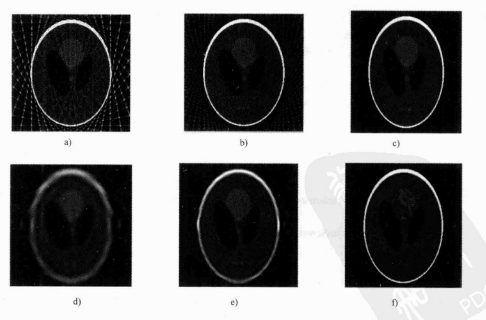


图 5-40　依不同投影密度重建的图像

a) I1（18）　b) I2（36）　c) I3（90）　d)Ifan1（2 弧度）　e) Ifan2（1 弧度）　f) Ifan3（0.25 弧度）

习题

5-1 离散傅里叶变换的性质及在图像处理中的应用有哪些？

5-2 求如图 5-41 所示图像的二维离散傅里叶变换。

① 长方形图像

$$f(x,y)=\begin{cases}E & |x|<a,|y|<b \\ 0 & \text{其他}\end{cases}$$

② 旋转 45° 后的长方形图像。

5-3 构造 $N=8$ 的 Haar 变换矩阵。

5-4 请用 C 语言或者 MATLAB 编程做出如图 5-42 所示图像的二维离散余弦变换。

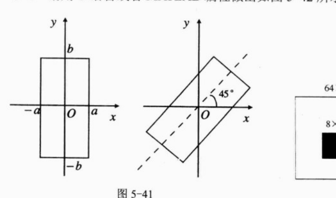


图 5-41

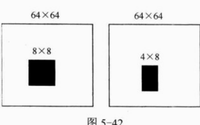


图 5-42

5-5 编写一个程序，要求实现下列算法：首先将图像分割为 8×8 的子图像，对每个子图像进行 FFT，对每个子图像中的 64 个系数，按照每个系数的方差来排序后，舍去小的变换系数，只保留 16 个系数，实现 4∶1 的图像压缩。

5-6 离散哈达玛变换的最大优点是什么？

5-7 K-L 变换的优点是什么？K-L 变换都有哪些性质？

5-8 求下面离散图像矩阵的二维离散傅里叶变换、离散沃尔什变换和离散哈达玛变换。

$$\begin{pmatrix} 0 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \end{pmatrix}$$

5-9 给定一幅行和列都为 2 的整数次幂图像，用 Haar 小波基函数对其进行二维小波变换，试着将最低尺度近似分量 W_φ 置零再反变换，结果是什么？如果把垂直方向的细节分量置零，反变换后结果又是什么呢？试解释一下原因。

5-10 小波变换是如何定义的？小波分析的主要优点是什么？

第6章　图像处理中的代数运算及几何变换

图像处理是建立在各种算法基础上的处理方法，本章围绕数字图像处理中的基本运算，主要介绍图像处理中的点运算、代数运算和几何变换，及其在图像处理中的应用等。

6.1　基本运算类型

在数字图像处理中，经常需要采用各种各样的算法，根据数字图像处理运算中输入信息与输出信息的类型，具有代表性的图像处理典型算法从功能上包括以下几种：

1）单幅图像→单幅图像。

2）多幅图像→单幅图像。

3）单幅图像或多幅图像→数值/符号等。

以上三类运算形式中，所有输入信息都是图像且其灰度值都是非负整数值，而输出信息的形式则各不相同，既可以是具有非负灰度值的数字图像，又可以是仅具有 0、1 两个灰度值的二值图像，也可以是对输入图像逐个像素点进行解释的符号或由特定参数组成的某种二维信息（又称为标号图像），还可以是从图像中提取出的以数值或符号描述的特征信息。所有以二维信息形式输出的信息统称为广义图像，标号图像也属于广义图像的范畴。

在三类运算中，第一类运算功能是数字图像处理技术中最基本的功能。对基本的图像处理功能，根据输入图像得到输出图像（目标图像）处理运算的数学特征，可将图像处理运算方式分为点运算、代数运算和几何运算。这些运算都是基于空间域的图像处理运算，与空间域运算相对应的是变换域运算。

6.2　点运算

在图像处理运算中，点运算（Point Operation）是一类简单却非常具有代表性的重要算法之一，也是其他图像处理运算的基础。运用点运算可以改变图像数据所占据的灰度值范围。对于一幅输入图像，经过点运算会产生一幅输出图像，输出图像中每个像素点的灰度值仅由相应输入像素点的灰度值确定。这与领域处理算法截然不同，在领域处理算法中，每个输出像素的灰度值由对应输入像素的一个领域内若干像素点的灰度值共同决定。因此，点运算不会改变图像内的空间位置关系。点运算是图像数字化软件以及图像显示软件的重要组成部分。

6.2.1　点运算的种类

点运算从数学上可以分为线性点运算和非线性点运算两类。

1．线性点运算

线性点运算是指输入图像的灰度级与目标图像的灰度级呈线性关系。线性点运算的灰度

变换函数形式可以采用线性方程描述，即

$$D_B = aD_A + b \tag{6-1}$$

式中，D_A 为输入点的灰度值；D_B 为相应输出点的灰度值。这种线性运算关系如图 6-1 所示。

2．非线性点运算

除了线性点运算外，还有非线性点运算。一般考虑非减（Nondecreasing）的灰度变换函数，其灰度变换关系如图 6-2 所示。非线性点运算的灰度变换函数的斜率处处为正数，这类函数保留了图像的基本外貌。

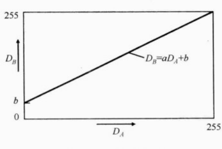


图 6-1 线性点运算

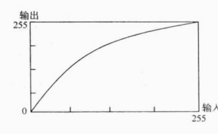


图 6-2 非线性点运算

非线性点运算的函数形式可以表示为

$$D_B = f(D_A) \tag{6-2}$$

式中，D_A 为输入点的灰度值；D_B 为相应输出点的灰度值；f 表示非线性函数，函数表达式须根据具体应用选择有代表性的非线性函数形式。

以下是三种典型的非线性点运算函数。

$$D_B = f(D_A) = D_A + CD_A(D_m - D_A) \tag{6-3}$$

式中，D_m 为灰度级的最大值；参数 C 定义了中间灰度范围内的增加量（$C>0$）或减少量（$C<0$）。非线性点运算可增加中间范围像素的灰度级而只使暗像素和亮像素进行较小改变。

$$f(D_A) = \frac{D_m}{2}\left\{1 + \frac{1}{\sin\left(\frac{a\pi}{2}\right)}\sin\left[a\pi\left(\frac{D_A}{D_m} - \frac{1}{2}\right)\right]\right\} \quad 0 < a < 1 \tag{6-4}$$

式中，灰度级范围为 0～D_m，在该灰度级范围内，直方图非零。参数 a 越大，效果越明显。该函数是基于正弦函数的非线性点运算形式。这类非线性单调点运算通过降低较高或较暗物体的对比度来加强灰度级处于中间范围的物体的对比度，这类函数曲线为 s 形，其灰度变换函数在中间部分的斜率大于 1，而两端处斜率小于 1。

$$f(D_A) = \frac{D_m}{2}\left\{1 + \frac{1}{\tan\left(\frac{a\pi}{2}\right)}\tan\left[a\pi\left(\frac{D_A}{D_m} - \frac{1}{2}\right)\right]\right\} \quad 0 < a < 1 \tag{6-5}$$

该函数是基于正切函数的非线性点运算形式。同样，参数a决定点运算的效果。这类非线性单调点运算是通过降低灰度级处于中间范围的物体的对比度，而将较亮和较暗物体的对比度加强。该灰度变换函数在中间处的斜率小于1，而在靠近两端处斜率大于1。

6.2.2 点运算与直方图

点运算是一种在确定的函数关系下所进行的像素变换运算，因此，点运算之后输出图像和输入图像之间的直方图也具有与变换函数相关联的对应关系。

设点运算之间的函数关系为$D_B = f(D_A)$，输入直方图、灰度变换函数以及输出直方图之间的关系如图 6-3 所示。灰度值D_A转换为D_B，同样，灰度值$D_A + \Delta D_A$转换为$D_B + \Delta D_B$，且在灰度级$[D_A, D_A + \Delta D_A]$之间的所有像素被转化到灰度级$[D_B, D_B + \Delta D_B]$之间，进一步可得出，灰度级$[D_A, D_A + \Delta D_A]$之间的输入像素的个数等于灰度级$[D_B, D_B + \Delta D_B]$之间的输出像素的个数，即

$$\int_{D_A}^{D_A+\Delta D_A} H_A(D)\mathrm{d}D = \int_{D_B}^{D_B+\Delta D_B} H_B(D)\mathrm{d}D \tag{6-6}$$

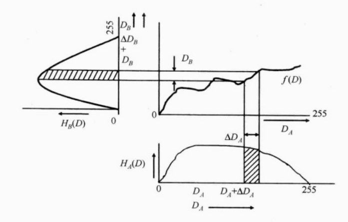


图 6-3　点运算与直方图之间的关系

当ΔD_A很小时，ΔD_B的取值也较小，因此，输入、输出图像阴影部分的面积可分别用两个小矩形的面积近似替代。

$$H_B(D_B)\Delta D_B = H_A(D_A)\Delta D_A \tag{6-7}$$

因此，输出直方图的值为

$$H_B(D_B) = H_A(D_A)\Delta D_A / \Delta D_B = \frac{H_A(D_A)}{\Delta D_B / \Delta D_A} \tag{6-8}$$

若ΔD_A趋于0，则取极限，因而可用微分表示：

$$H_B(D_B)=\frac{H_A(D_A)}{\frac{\mathrm{d}D_B}{\mathrm{d}D_A}} \tag{6-9}$$

代入点运算的函数形式得如下结论：

$$H_B(D_B)=\frac{H_A(D_A)}{\frac{\mathrm{d}f(D_A)}{\mathrm{d}D_A}} \tag{6-10}$$

6.2.3 点运算的应用

由于点运算能有规律地改变像素点的灰度值，因而点运算有时又被称为对比度增强或灰度变换（Gray-Scale Transformation，GST）。因此，通过恰当定义数学运算的形式，点运算可用于改善图像数字化设备或图像数字显示设备的某些局限性。

1. 对比度增强

在一些数字图像中，技术人员所关注的特征可能仅占整个灰度级范围非常小的一部分。点运算可以扩展所关注部分的灰度信息的对比度，使其占据可显示灰度级的更大部分。该方法有时被称为对比度增强（Contrast Enhancement）或对比度拉伸（Contranst Stretching）。

2. 光度学标定

人们常常希望数字图像的灰度能反映诸如光照强度、光密度等某些物理特性，通过去掉图像传感器的非线性影响，点运算可达到该目的。例如，假设一幅图像被一个对光照强度呈非线性关系的仪器所数字化，点运算可以通过适当的灰度变换运算，使灰度级与光照强度的等步长增量匹配。

点运算的另一个用处是变换灰度的单位。假定有一个图像数字化仪器，用来数字化一幅显微镜下观察到的图像。其产生的灰度值与标本的透射率呈线性关系，点运算可用来产生一幅图像，该图像的灰度级可代表光学密度的等步长增量。光度学标定通常作为图像数字化的软件部分。

3. 显示标定

一些显示设备通常具有能突出图像视觉特征的优选灰度范围。使用这样的显示设备时，数字图像中具有相同对比度的较暗和较亮的特征，在显示时却不能以同样的性能表现出来。在这种情况下，用户可以利用点运算让感兴趣的所有特征同等突出地显示出来。

一些显示设备不能保持数字图像上像素的灰度值和显示屏幕上相应点的亮度之间的线性关系，同样，许多胶片记录仪不能线性地将灰度值转换为光密度。这一缺点也可以通过点运算予以克服，即在图像显示之前，先设计合理的点运算关系。另外，可将点运算和显示非线性组合起来互相抵消，以保持在显示图像时的线性关系。这一过程就是显示标定（Display Calibration）。

少数情况下，非线性显示关系对于图像的表示也具有一定的作用。例如，电视机或 CRT 显示器的γ校正就是利用了这种非线性关系，点运算可纠正或调整显示的γ值。

点运算有时被视为强化细节或增加图像某些部分的对比度的图像处理步骤。然而，由于信息实际上包含在数字图像中，所以，实际要做的工作是使感兴趣部分的灰度级与显示设备的对比度范围匹配起来。显示标定和对比度增强也经常作为数字图像显示的软件部分。

4．轮廓线

点运算可为图像加上轮廓线。可以应用点运算进行阈值化，根据灰度级可将一幅图像划分成一些不连接的区域，有助于在后续处理中确定边界或用于定义掩膜。

5．剪裁

因为数字图像通常以整型格式存储，所以，可用的灰度级范围是有限的。对于 8bit 图像，在每个像素值被存储之前，输出灰度级一定要被裁剪到 0～255 的范围内。

6.3 图像的代数运算

图像的代数运算是指对两幅或两幅以上输入图像的对应像素逐个进行加、减、乘、除四则运算，以产生有增强效果的图像。图像的代数运算是一种比较简单和有效的增强处理，是图像增强处理中的常用方法。为此，MATLAB 7.0 的图像处理工具箱提供了一套图像代数运算函数，使得对图像的代数运算变得非常容易。

图像代数运算函数可以处理包括 uint8、uint16 和 double 等各种类型在内的数值数据，并返回相同类型的结果图像。

6.3.1 图像代数的异常处理

图像数据不同于一般意义的数据，在执行代数运算得到结果图像的时候，必须注意图像数据的物理意义，保证计算结果的合理性。否则，在执行图像代数运算时，结果经常会出现一些异常情况。常见的异常情况有以下两种。

（1）计算结果溢出

很多图像，如灰度图像、索引色图像、二值图像或有限位真彩色图像，其像素值是有范围限制的，然而在执行两幅或多幅图像的加、减或乘法运算时，计算结果很可能会超出限定的有效范围，如两幅 256 色灰度图像在执行减法运算时，很可能会出现像素值为负值的情况。或者执行加法和乘法运算时，像素值超过 255，这都是异常的结果，必须改正。

（2）计算结果类型无效

图像数据有多种存储类型，如 uint8 或 uint16，像素值要求是整数类型，然而在进行除法运算时，往往会得到分数的计算结果，这是因为图像代数运算作为函数在执行运算时，把图像数据看为 double 类型。这是另一种异常的图像代数运算结果，也必须加以改正。

MATLAB 中用于普通代数运算的操作符尽管也可以执行加、减、乘、除四则运算，但它们对计算结果的有效性不予检查，直接以实数运算的结果进行表示。然而，用于图像的代数运算函数则会自动地对计算结果进行有效修正。下面分别介绍一下对以上出现的两种异常结果的修正方式。

异常计算结果的修正遵循两个原则：

1）超过整数类型有效范围的结果直接截断到限定范围的端点值。

2）对于分数计算结果采取四舍五入。

例如，如果被处理的图像数据是 uint8 类型的，当计算结果出现以下情况时的修正结果见表 6-1。

表 6-1 像素值取整原则等修正举例

理论计算结果	函数返回值类型	实际输出结果
300	uint8	255
-30	uint8	0
10.5	uint8	11

类似于一般的四则运算，我们可以嵌套使用图像代数运算函数，即组合多个图像代数运算函数来完成一系列操作。例如，要计算两幅图像的平均值，用户通常会想到下面的几行代码：

```
I=imread('rice.png');
I2=imread('cameraman.tif');
K=imdivide(imadd(I, I2), 2);
```

上面的前两条语句是分别读取图像 rice.png 和图像 cameraman.tif 到变量 I 和 I2，第三条语句是组合使用加法函数 imadd()和除法函数 imdivide()，分析起来，这样的代码完全可以完成预定的操作。然而，当我们读进来的图像数据是 uint8 或 uint16 类型时，图像代数运算函数会自动地按照上面介绍的两个原则对结果进行修正，而且 MATLAB 图像代数运算函数是每执行一次代数运算就执行一次修正，这样对于第三条语句的嵌套调用则是先对加法运算的结果进行修正再对除法运算的结果进行修正，这就显著地减少了结果图像中包含的大量信息。为了能够得到更好的结果，在嵌套调用图像代数运算函数时，可以考虑使用函数 imlincomb()，这个函数采用线性组合的方式按照 double 类型执行所有的图像代数运算函数，而且只对最终结果进行数据有效性修正。为此，上面的两幅图像求平均运算，可以采用如下更合适的语句：

```
K=imlincomb(.5,I,.5,I2);
```

6.3.2 各种代数运算

1. 加法运算

图像加法运算一般用于对同一场景的多幅图像求平均效果，以便有效地降低具有叠加性质的随机噪声。直接采集的图像品质一般都较好不需要进行加法运算处理，但是对于那些经过长距离模拟通信方式传送的图像（如卫星图像），这种处理是必不可少的。

在 MATLAB 7.0 中，如果要进行两幅图像的加法，或者给一幅图像加上一个常数，可以调用 imadd()函数来实现。imadd()函数将某一输入图像的每一像素值与另一幅图像相应的像素值相加，返回相应的像素值之和作为输出图像的对应像素值。imadd()函数的调用格式可参考图像 MATLAB 7.0 处理工具箱。下面的程序可将如图 6-4a、b 所示的图像叠加在一起生成新图像，如图 6-4c 所示。



a)



b)



c)

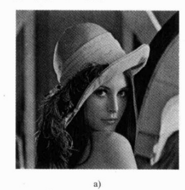
图 6-4 图像相加运算

a) 原始图像 rice.png b) 原始图像 cameraman.tif c) 加法运算后的图像

```
I=imread('rice.png');
figure(1),imshow(I,[])
I2=imread('cameraman.tif')
figure(2),imshow(I2,[])
K=imadd(I,I2,'uint16');
figure(3),imshow(K,[])
```

给图像的每一个像素加上一个常数可以使图像的整体亮度增加。例如，以下程序实例的处理效果如图 6-5 所示。

```
I=imread(' F:\image \lena. bmp');
I2=imadd(I,50);
figure(1),imshow(I)
figure(2),imshow(I2)
```



a)



b)

图 6-5 图像亮度整体增强

a) 原始图像 lena.bmp b) 加入常数后的图像

对两幅图像的像素值进行相加操作，其结果很可能超过图像数据类型所支持的最大值，尤其对于 uint8 类型的图像，溢出情况最为常见。当数据值发生溢出时，imadd()函数会将数据截取为数据类型所支持的最大值，这种截取效果称为饱和处理。为了避免出现饱和现象，

在进行加法计算前最好将图像转换为一种数据范围较宽的数据类型。例如，在加法操作前将uint8 类型的图像转换为 uint16 类型。

2. 减法运算

图像减法也称为差分方法，是一种常用于检测图像变化及运动物体的图像处理方法。图像减法可以作为许多图像处理过程的准备步骤。例如，可以使用图像减法来检测一系列相同场景图像的差异。利用图像减法处理图像时，往往需要考虑背景的更新机制，尽量补偿因天气、光照等因素对图像显示效果造成的影响。

在 MATLAB 7.0 中，使用 imsubtract()函数可以将一幅图像从另一幅图像中减去，或者从一幅图像中减去一个常数。imsubtract()函数将一幅输入图像的像素值从另一幅输入图像相应的像素值中减去，再将相应的像素值之差作为输出图像相应的像素值。执行两幅图像相减操作，生成如图 6-6 所示的图像，程序代码如下：

```
I=imread('rice.png');
I2=imread('cameraman.tif')
I3=imsubtract(I,I2);
imview(I3)
```

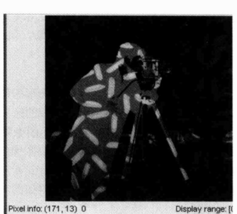
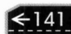


图 6-6　图像减法运算

3. 乘法运算

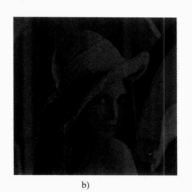
两幅图像进行乘法运算可以实现掩模操作，即屏蔽掉图像的某些部分。一幅图像乘以一个常数通常被称为缩放，这是一种常见的图像处理操作。如果使用的缩放因数大于 1，那么将增强图像的亮度；如果缩放因数小于 1，则会使图像变暗。缩放操作通常会产生比简单添加像素偏移量更自然的明暗效果。这是因为该操作能够更好地维持图像的相关对比度。此外，由于时频的卷积或相关运算与频域的乘积运算对应，因此乘法运算有时也成为一种技巧，用来实现卷积或相关处理。

在 MATLAB 7.0 中，可以使用 immultiply( )函数实现两幅图像的乘法或一幅图像的亮度缩放，immultiply()函数将两幅图像相应的像素值进行元素对元素的乘法操作，即图像矩阵的点乘运算，并将乘法的结果作为输出图像相应的像素值。例如，以下程序实例将使用给定的缩放因数对如图 6-7a 所示的图像进行亮度缩放，从而得到如图 6-7b 所示的较为暗淡的图像。





a)



b)

图 6-7　图像乘法运算

a) 原始图像　b) 相乘后的图像

```
I=imread('lena.bmp');
I2=immultiply(I,0.5);
figure,imshow(I);
figure,imshow(I2)
```

Uint8 类型的图像在进行乘法运算时，一般都会发生溢出现象。immultiply()函数将溢出的数据截取为数据类型的最大值。为了避免产生溢出现象，可以在执行乘法运算之前将 uint8 类型的图像转换为一种数据范围较大的图像类型。

4．除法运算

除法运算可用于校正成像设备的非线性影响，这在特殊形态的图像（如断层扫描等医学图像）处理中经常用到。图像除法也可以用来检测两幅图像的区别，但是除法操作给出的是相应像素值的变换比率，而不是每个像素的绝对差异，因而图像除法操作也称为比率变换。

在 MATLAB 7.0 中，可以使用 imdivide()函数进行两幅图像的除法运算或一幅图像的亮度缩放。imdivide 函数对两幅输入图像的所有相应像素执行元素对元素的除法运算，并将得到的结果作为输出图像的相应像素值。以下程序代码实例将如图 6-8a 所示的图像进行除法运算，得到如图 6-8b 所示的效果。



a)



b)

图 6-8　图像除法运算

a) 原始图像 lena.bmp　b) 除去一个常数后的效果

```
I=imread('lena.bmp');
I2=imdivide(I,0.5);
figure,imshow(I);
figure,imshow(I2)
```

6.4 几何变换基础

6.4.1 齐次坐标

数字图像是把连续图像在坐标空间和性质空间离散化了的图像。例如，一幅二维数字图像可以用一组二维(2D)数组 $f(x,y)$ 来表示，其中 x 和 y 表示 $2D$ 空间 xy 中一个坐标点的位置，$f(x,y)$ 代表图像在点 (x,y) 的某种性质的数值。如果所处理的是一幅灰度图像，这时 $f(x,y)$ 表示灰度值，此时 $f(x,y)$，x,y 都在整数集合中取值。因此，除了插值运算外，常见的图像几何变换可以通过与其对应的矩阵线性变换来实现。

设点 $P_0(x_0,y_0)$ 进行平移后，移到 $P(x,y)$，其中 x 方向的平移量为 Δx，y 方向的平移量为 Δy，如图 6-9 所示，那么，点 $P(x,y)$ 的坐标为

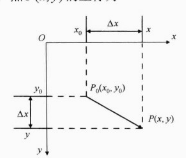


图 6-9 图像的平移变换示意图

$$\begin{cases} x = x_0 + \Delta x \\ y = y_0 + \Delta y \end{cases} \quad (6\text{-}11)$$

这个变换矩阵的形式可以表示为

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \quad (6\text{-}12)$$

对式（6-12）进行简单变换可得

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \quad (6\text{-}13)$$

对式（6-13）进一步变换，可得

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \tag{6-14}$$

式（6-14）中等号右侧左面矩阵的第 1、2 列构成单位矩阵，第 3 列元素为平移常量。该矩阵是点 $P_0(x_0, y_0)$ 平移到点 $P(x, y)$ 的平移矩阵，即为变换矩阵。该变换矩阵是 2×3 阶的矩阵，为了符合矩阵相乘时要求前者列数与后者行数相等的规则，需要在点的坐标列矩阵 $(x_0 \quad y_0)^{\mathrm{T}}$ 中引入第 3 个元素，增加一个附加坐标，扩展为 3×1 阶的列矩阵 $(x_0 \quad y_0 \quad 1)^{\mathrm{T}}$。这样，式（6-13）同式（6-14）表述的意义完全相同。为了使式（6-14）左侧表示成矩阵 $(x \quad y \quad 1)^{\mathrm{T}}$ 的形式，可用三维空间点 $(x, y, 1)$ 表示二维空间点 (x, y)，即采用一种特殊的坐标，可以实现平移变换，变换结果如下：

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \tag{6-15}$$

现对式（6-15）中的各个矩阵进行定义：

$$\boldsymbol{T} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \text{为变换矩阵；}$$

$$\boldsymbol{P} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \text{为变换后的坐标矩阵；}$$

$$\boldsymbol{P}_0 = \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \text{为变换前的坐标矩阵；}$$

则有

$$\boldsymbol{P} = \boldsymbol{T}\boldsymbol{P}_0 \tag{6-16}$$

从式（6-16）可以看出，引入附加坐标后，扩充了矩阵的第 3 行，但并没有使变换结果受到影响。这种用 $n+1$ 维向量表示 n 维向量的方法称为齐次坐标表示法。

6.4.2 齐次坐标的一般表现形式及意义

1．齐次坐标的一般表现形式

式（6-16）给出了图像经过平移后齐次坐标的特殊形式，齐次坐标的一般形式可表示为

$$P=\begin{pmatrix}H_x\\H_y\\H\end{pmatrix} \tag{6-17}$$

式中，H 为非零的实数。当 H=1 时，$P=\begin{pmatrix}x & y & 1\end{pmatrix}^{\mathrm{T}}$ 称为规范化齐次坐标。

由点的齐次坐标 (H_x,H_y,H)，求点的规范化坐标 $(x,y,1)$。可按如下公式进行：

$$x=\frac{\boldsymbol{H}_x}{H} \qquad y=\frac{\boldsymbol{H}_y}{H} \tag{6-18}$$

2. 齐次坐标的意义

齐次坐标的几何意义相当于点 (x,y) 落在3D空间 $H=1$ 的平面上，如图 6-10 所示。如果将 xy 平面内的 $\triangle ABC$ 的各顶点表示成齐次坐标 $(x_i,y_i,1)$ $(i=1,2,3)$ 的形式，就变成 $H=1$ 平面内的 $\triangle A_1B_1C_1$ 的各顶点。

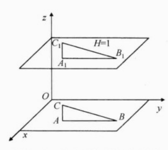


图 6-10 齐次坐标的几何意义

6.4.3 二维图像几何变换的矩阵

为了将式（6-16）写成一般形式，对包括平移在内的所有几何变换都适用，对 $\boldsymbol{P}$，$\boldsymbol{T}$，$\boldsymbol{P}_0$ 进行重新定义，齐次坐标为

$$\boldsymbol{P}=\begin{pmatrix}Hx_1 & Hx_2 & \cdots & Hx_n\\Hy_1 & Hy_2 & \cdots & Hy_n\\H & H & \cdots & H\end{pmatrix} \tag{6-19}$$

当 $H=1$ 时，规范化的齐次坐标为

$$\boldsymbol{P}=\begin{pmatrix}x_1 & x_2 & \cdots & x_n\\y_1 & y_2 & \cdots & y_n\\1 & 1 & \cdots & 1\end{pmatrix} \tag{6-20}$$

变换矩阵 $\boldsymbol{T}$ 为

$$\boldsymbol{T}=\begin{pmatrix}a & b & p\\c & d & q\\l & m & s\end{pmatrix} \tag{6-21}$$

$$\boldsymbol{P}_0=\begin{pmatrix} x_{01} & x_{02} & \cdots & x_{0n} \\ y_{01} & y_{02} & \cdots & y_{0n} \\ 1 & 1 & \cdots & 1 \end{pmatrix} \tag{6-22}$$

则上述变换可以用公式表示为

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ 1 & 1 & \cdots & 1 \end{pmatrix}=\boldsymbol{T}\begin{pmatrix} x_{01} & x_{02} & \cdots & x_{0n} \\ y_{01} & y_{02} & \cdots & y_{0n} \\ 1 & 1 & \cdots & 1 \end{pmatrix} \tag{6-23}$$

引入齐次坐标后，表示2D图像几何变换的3×3阶矩阵的功能就完善了，可以用它完成2D图像的各种几何变换。下面讨论3×3阶变换矩阵中元素在变换中的功能。几何变换的3×3阶矩阵的一般形式见式（6-21）。

3×3阶矩阵 $\boldsymbol{T}$ 可以分成 4 个子矩阵：$\begin{pmatrix} a & b \\ c & d \end{pmatrix}_{2\times2}$ 子矩阵可使图像实现恒等、比例、镜像、错切和旋转变换；$(p \quad q)^{\mathrm{T}}$ 列矩阵可以使图像实现平移变换；$(l \quad m)$ 行矩阵可以使图像实现透视变换，但当 $l=0$，$m=0$ 时，它无透视作用；s 这一元素可以使图像实现全比例变换。例如，将图像进行全比例变换，即

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{pmatrix}\begin{pmatrix} x_{0i} \\ y_{0i} \\ 1 \end{pmatrix}=\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{6-24}$$

由式（6-24）可知，当 $s\neq1$ 时，等式两端不相等，若想等式成立，应将式（6-24）变为

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{pmatrix}\begin{pmatrix} x_{0i} \\ y_{0i} \\ 1 \end{pmatrix}=\begin{pmatrix} x_{0i} \\ y_{0i} \\ 1 \end{pmatrix} \tag{6-25}$$

齐次坐标规范化后得

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{pmatrix}\begin{pmatrix} \frac{x_{0i}}{s} \\ \frac{y_{0i}}{s} \\ \frac{1}{s} \end{pmatrix}=\begin{pmatrix} x_i \\ y_i \\ s \end{pmatrix} \tag{6-26}$$

由式（6-26）可见，当 $s>1$ 时，图像按比例缩小，如 $s=2$，$x=\frac{x_0}{2}$，$y=\frac{y_0}{2}$，图像缩小到原来的 1/2；当 $0<s<1$ 时，整个图像按比例放大，如 $s=1/2$，$x=2x_0$，$y=2y_0$，图像放大到原来的 2 倍；当 $s=1$ 时，$x=x_0$，$y=y_0$，图像大小不变。

6.5 各种几何变换

6.5.1 图像平移变换

1. z图像平移变换

平移是日常生活中最普遍的方式之一，如开学时教室里课桌的重新摆放等都可以视为平移运动。图像的平移是将一幅图像上的所有像素点都按给定的偏移量沿 x 方向（水平方向）和 y 方向（垂直方向）进行移动，如图 6-11 所示。图像的平移变换是图像几何变换中最简单的变换之一。

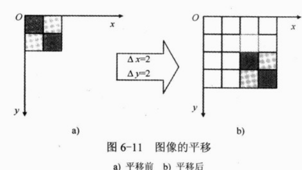


图 6-11 图像的平移

a) 平移前 b) 平移后

若点 $A_0(x_0,y_0)$ 进行平移后，被移到 $A(x,y)$ 位置，其中，x 方向上的平移量为 Δx，y 方向上的平移量为 Δy，那么，点 $A(x,y)$ 的坐标为（此公式 6.4 节已经列出）：

$$\begin{cases} x = x_0 + \Delta x \\ y = y_0 + \Delta y \end{cases}$$

利用齐次坐标，点 $A(x,y)$ 的坐标可以表示如下（此公式 6.4 节已经列出）：

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

相应地，也可以根据点 $A(x,y)$ 求解原始点 $A_0(x_0,y_0)$ 的坐标，即

$$\begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -\Delta x \\ 0 & 1 & -\Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{6-27}$$

显然，以上两个变换矩阵互为逆矩阵。

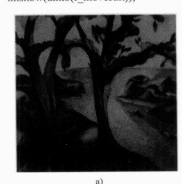
图像平移变换的特点是平移后的图像与原图像完全相同，平移后新图像上的每一点都可以在原图像中找到对应的点。仍以图 6-11 为例，对于新图像中的像素点（0，0），代入式（6-27）可以求出对应原始图像中的像素点（−2，−2），该点不在原始图像中。对于不在原始图像中的点，可以直接将它们的像素值统一设置为 0 或 255，对于灰度图像则为黑色或白色。反

之，若某像素点不在新图像中，同样说明原始图像中有某些像素点被移出了显示区域。图像经平移后，原始图像的一些像素点被移出了显示区域，若想保留全部图像，则应扩大新图像的显示区域。

2．图像平移变换的 MATLAB 实现

下面是图像平移的程序代码实例（见图 6-12）：

```
I=imread('trees.tif');
figure,imshow(I);
I=double(I);
I_movesult=zeros(size(I));
H=size(I);
Move_x=50;
Move_y=50;
I_movesult(Move_x+1:H(1),Move_y+1:H(2),1:H(3))=I(1:H(1)-Move_x,1:H(2)-Move_y,1:H(3));
imshow(uint8(I_movesult));
```


a)

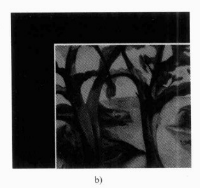

b)

图 6-12　图像的平移

a) 原始图像　b) 平移后的图像

6.5.2 图像比例变换

1．图像比例变换

通常情况下，数字图像的比例缩放是指给定的图像在 x 方向和 y 方向按相同的比例缩放 a 倍，从而获得一幅新的图像，又称为全比例缩放。如果 x 方向和 y 方向缩放的比例不同，则图像的比例缩放会改变原始图像像素间的相对位置，产生几何畸变。设原始图像中的点 $A_0(x_0,y_0)$ 比例缩放后，在新图像中的对应点为 $A_1(x_1,y_1)$，则 $A_0(x_0,y_0)$ 和 $A_1(x_1,y_1)$ 之间的坐标关系可表示为

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \boldsymbol{T} = \begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \tag{6-28}$$

即

$$\begin{cases} x_1 = ax_0 \\ y_1 = ay_0 \end{cases}$$

若比例缩放所产生的图像中的像素在原始图像中没有相对应的像素点时，就需要进行灰度值的插值运算，一般有以下两种插值处理方法。

1）直接赋值为和它最相近的像素灰度值，这种方法称为最邻近插值法（Nearest Neighbor Interpolation），该方法的主要特点是简单、计算量很小，但可能会产生马赛克现象。

2）通过其他数学插值算法来计算相应的像素点的灰度值，这类方法处理效果好，但运算量会有所增加。

在式（6-28）所表示的比例缩放中，若 $a>1$，则图像被放大；若 $a<1$，则图像被缩小。以 $a=1/2$ 为例，即图像被缩小为原始图像的一半。图像被缩小一半以后根据目标图像和原始图像像素之间的关系，有如下两种缩小方法。

第一种方法如图 6-13 所示，此时缩放前后图像间像素点的对应关系如下：

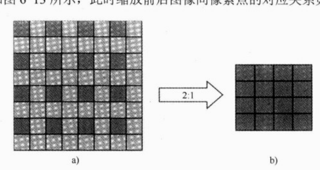


图 6-13　图像缩小一半（偶数行和偶数列构成新的图像）

a) 原始图像　b) 缩小后的图像

$$\text{缩小图像}\begin{cases} (0,0) \leftrightarrow (0,0) \\ (0,1) \leftrightarrow (0,2) \\ (0,2) \leftrightarrow (0,4) \\ (0,3) \leftrightarrow (0,6) \\ (1,0) \leftrightarrow (2,0) \\ (2,0) \leftrightarrow (4,0) \\ \quad\vdots \qquad\quad \vdots \\ (3,0) \leftrightarrow (6,0) \\ (3,1) \leftrightarrow (6,2) \\ (3,2) \leftrightarrow (6,4) \\ (3,3) \leftrightarrow (6,6) \end{cases}\text{原始图像}$$

依此类推，可以逐点计算缩小图像各像素点的值，图像缩小之后所承载的信息量为原始图像的 50%，即在原始图像上，按行优先的原则，对所处理的行，每隔一个像素取一点，每隔一行进行一次操作。也就是说，取原始图像的偶数行和偶数列构成新的图像。

另一种方法是取原始图像的奇数行和奇数列组成新的图像，如图 6-14 所示。

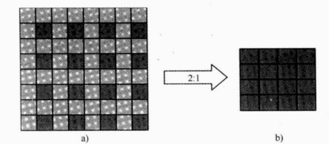


图 6-14　图像缩小一半（奇数行和奇数列构成新的图像）

a) 原始图像　b) 缩小后的图像

如果图像按任意比例缩小，则以类似的方式按比例选择行和列上的像素点。若 x 方向与 y 方向缩放比例不同，则这种变换将会使缩放以后的图像产生几何畸变。图像 x 方向与 y 方向的不同比例缩放的变换公式如下：

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \boldsymbol{T} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \quad a \neq b \tag{6-29}$$

图像缩小变换是在已知的图像信息中以某种方式选择需要保留的信息。反之，图像的放大变换则需要对图像尺寸经放大后所多出来的像素点填入适当的像素值，这些像素点在原始图像中没有直接对应点，需要以某种方法进行估计。以 $a=b=2$ 为例，即原始图像按全比例放大 2 倍，实际上，这是将原始图像每行中各像素点重复取一遍值，然后每行重复一次。根据理论计算，放大以后图像中的像素点（0,0）对应于原始图像中的像素点（0,0），（0,2）对应于原始图像中的（0,1），但放大后图像的像素点（0,1）对应于原始图像中的像素点（0,0.5），（1,0）对应于原始图像中的（0.5,0），原始图像中不存在这些像素点，那么放大后的图像如何处理这些问题呢？以像素点（0,0.5）为例，这时可以采取以下两种方法和原始图像对应，其余点依此逐点类推。

1）将原始图像中的像素点（0,0.5）近似为原始图像的像素点（0,0）。

2）将原始图像中的像素点（0,0.5）近似为原始图像的像素点（0,1）。

以图 6-15 所示的一段直线为例来说明这两种放大的细微差别。

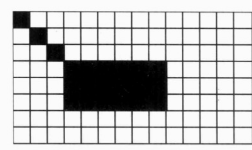


图 6-15　放大前的原始图像

若采取第 1 种方法，则原始图像和放大图像的像素点对应关系如下：

$$
\text{放大图像}\left\{\begin{matrix}
(0,0)\leftrightarrow(0,0)\\
(0,1)\leftrightarrow(0,0)\\
(1,0)\leftrightarrow(0,0)\\
(1,1)\leftrightarrow(0,0)\\
(2,2)\leftrightarrow(1,1)\\
(2,3)\leftrightarrow(1,1)\\
(3,2)\leftrightarrow(1,1)\\
(3,3)\leftrightarrow(1,1)\\
(4,4)\leftrightarrow(2,2)\\
(4,5)\leftrightarrow(2,2)\\
(5,4)\leftrightarrow(2,2)\\
(5,5)\leftrightarrow(2,2)
\end{matrix}\right\}\text{原始图像}
$$

其对应的放大图像如图 6-16 所示。

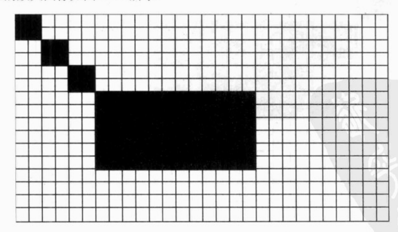


图 6-16　按最近邻域法放大两倍的图像

若采取第 2 种方法，则原始图像和放大图像的像素点对应关系如下：



$$
\text{放大图像}\left\{\begin{matrix}
(0,0)\leftrightarrow(0,0)\\
(0,1)\leftrightarrow(0,1)\\
(1,0)\leftrightarrow(1,0)\\
(1,1)\leftrightarrow(1,1)\\
(1,2)\leftrightarrow(1,1)\\
(2,1)\leftrightarrow(1,1)\\
(2,2)\leftrightarrow(1,1)\\
(2,3)\leftrightarrow(1,1)\\
(3,2)\leftrightarrow(1,1)\\
(3,3)\leftrightarrow(1,1)\\
(3,4)\leftrightarrow(2,2)\\
(4,3)\leftrightarrow(2,2)\\
(4,4)\leftrightarrow(2,2)\\
(5,5)\leftrightarrow(3,3)\\
(5,6)\leftrightarrow(3,3)\\
(6,3)\leftrightarrow(3,3)\\
(6,6)\leftrightarrow(3,3)
\end{matrix}\right\}\text{原始图像}
$$

其中对应的放大图像如图 6-17 所示。因此，两种放大方式具有一定差别。

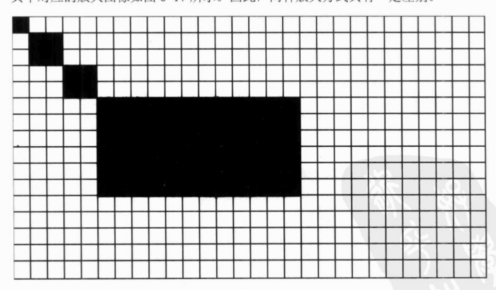


图 6-17　按插值方法放大两倍的图像

一般地，按比例将原始图像放大 a 倍时，如果按照最近邻域法，则需要将一个像素值添到新图像的 $a\times a$ 的方块中，如图 6-18 所示。因此，如果放大倍数过大，则按照这种方法填

充灰度值会出现马赛克效应。为了避免马赛克效应，提高几何变换后的图像质量，可以采用不同复杂程度的线性插值法填充放大后所多出来的相关像素点的灰度值。

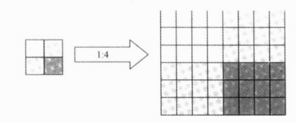


图 6-18　按最近邻域法放大四倍的图像

2. 图像比例变换的 MATLAB 实现

图像缩放的 MATLAB 应用，图像缩放函数的格式参考以下程序代码：

```
J=imread('trees.tif');
imshow(J);
X1=imresize(J,2) %放大为原来的两倍
X2=imresize(J,0.5) %缩小为来的 1/2
figure,imshow(X1);
figure,imshow(X2)
```

执行程序后结果如图6-19所示。



a)　　b)



c)

图 6-19　图像缩放

a) 原始图像　b) 放大后的图像　c) 缩小后的图像

6.5.3 图像旋转变换

1. 旋转变换

旋转（Rotation）有一个绕着什么转的问题。通常的做法是，以图像的中心为圆心旋转，将图像上的所有像素都旋转一个相同的角度。图像的旋转变换是图像的位置变换，但旋转后，图像的大小一般会改变。和图像平移变换一样，在图像旋转变换中，可以把转出显示

区域的图像截去，旋转后也可以扩大图像范围以显示所有的图像。图 6-20 是旋转前的图像；图 6-21 是将图 6-20 旋转 30°（顺时针方向）后保持原图像大小，转出的部分被裁掉的情况；图 6-22 是不裁掉转出部分，旋转后图像放大的情况。

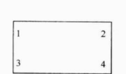


图 6-20　旋转前的图像

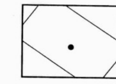


图 6-21　保持原图像大小的旋转

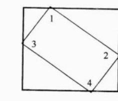


图 6-22　图像放大的旋转

采用不裁掉转出部分，旋转后图像变大的做法（见图 6-22），首先给出变换矩阵。在我们熟悉的坐标系中，如图 6-23 所示，将一个点顺时针旋转 a 角，r 为该点到原点的距离，b 为 r 与 x 轴之间的夹角。在旋转过程中，r 保持不变。

设旋转前 x_0，y_0 的坐标分别为 $x_0 = r\cos b$；$y_0 = r\sin b$。当旋转 a 角度后，坐标 x_1, y_1 的值分别为

$$
\begin{aligned}
x_1 &= r\cos(b-a) = r\cos b\cos a + r\sin b\sin a = x_0\cos a + y_0\sin a \\
y_1 &= r\sin(b-a) = r\sin b\cos a - r\cos b\sin a = -x_0\sin a + y_0\cos a
\end{aligned} \tag{6-30}
$$

以矩阵的形式表示为

$$
\begin{pmatrix} x_1 & y_1 & 1 \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix}\begin{pmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6-31}
$$

式（6-31）中，坐标系 xOy 是以图像的中心为原点，向右为 x 轴正方向，向上为 y 轴正方向。它与以图像左上角点为原点 O'，向右为 x' 轴正方向，向下为 y' 轴正方向的坐标系 $x'y'$ 之间的转换关系如图 6-24 所示。

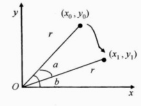


图 6-23　旋转示意图

图 6-24　两种坐标系间的转换关系

设图像的宽为 w，高为 h，容易得到

$$
\begin{pmatrix} x & y & 1 \end{pmatrix} = \begin{pmatrix} x' & y' & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w & 0.5h & 1 \end{pmatrix} \tag{6-32}
$$

逆变换为

$$\begin{pmatrix} x' & y' & 1 \end{pmatrix} = \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w & 0.5h & 1 \end{pmatrix} \tag{6-33}$$



有了式（6-31）～（6-33），可以将变换分成 3 个步骤来完成：

1）将坐标系 $xO'y$ 变成 xOy 。

2）将该点顺时针旋转 a 角。

3）将坐标系 xOy 变回 $xO'y$ ，这样，就得到了如下的变换矩阵（是上面 3 个矩阵的级联）。

$$\begin{aligned}\begin{pmatrix} x_1 & y_1 & 1 \end{pmatrix} &= \begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w_{\text{old}} & 0.5h_{\text{old}} & 1 \end{pmatrix} \begin{pmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w_{\text{new}} & 0.5h_{\text{new}} & 1 \end{pmatrix} \\ &= \begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{pmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ -0.5w_{\text{old}}\cos a + 0.5h_{\text{old}}\sin a - 0.5w_{\text{new}} & -0.5w_{\text{old}}\sin a - 0.5h_{\text{old}}\cos a + 0.5h_{\text{new}} & 1 \end{pmatrix}\end{aligned} \tag{6-34}$$

式中，w_{old}，h_{old}，w_{new}，h_{new} 分别表示原图像的宽、高和新图像的宽、高。式（6-34）的逆变换为

$$\begin{aligned}\begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} &= \begin{pmatrix} x_1 & y_1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w_{new} & 0.5h_{new} & 1 \end{pmatrix} \begin{pmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5w_{\text{old}} & 0.5h_{\text{old}} & 1 \end{pmatrix} \\ &= \begin{pmatrix} x_1 & y_1 & 1 \end{pmatrix} \begin{pmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ -0.5w_{\text{new}}\cos a - 0.5h_{\text{new}}\sin a + 0.5w_{\text{old}} & 0.5w_{\text{new}}\sin a - 0.5h_{\text{new}}\cos a + 0.5h_{\text{old}} & 1 \end{pmatrix}\end{aligned} \tag{6-35}$$

这样，对于新图像中的每一点，就可以根据式（6-34）求出对应原图像中的点，并得到它的灰度。如果超出原图像范围，则填成白色。要注意的是，由于有浮点运算，计算出来的点的坐标可能不是整数，需要采用取整处理，即找最接近的点，这样会带来一些误差（图像可能会出现锯齿）。更精确的方法是采用插值，这将在 6.6 节中进行介绍。

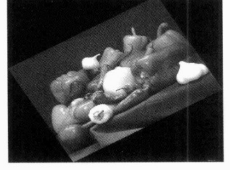
2. 图像旋转变换的 MATLAB 实现

图像分别逆时针旋转 30°、45°和 60°的程序代码如下，程序运行结果如图 6-25 所示。

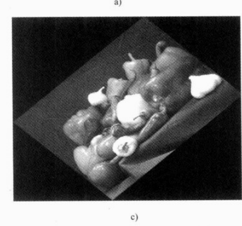
```
I=imread('peppers.png')
imshow(I);
X1=imrotate(I,30,'nearest'); %旋转 30°
figure;
imshow(uint8(X1));
figure;
X2=imrotate(I,45,'nearest'); %旋转 45°
imshow(uint8(X2));
figure;
X3=imrotate(I,60,'nearest'); %旋转 60°
imshow(uint8(X3));
```



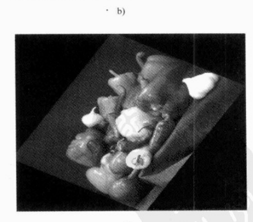
a)



b)



c)



d)

图 6-25 图像旋转变换

a) 原始图像 b) 旋转 30°后的图像 c) 旋转 45°后的图像 d) 旋转 60°后的图像

6.5.4 图像镜像变换

图像的镜像变换不改变图像的形状，图像的镜像（Mirror）变换分为 3 种：水平镜像、垂直镜像和对角镜像。

1. 图像的水平镜像

设图像的水平镜像操作是将图像左半部分和右半部分以图像垂直中轴线为中心进行镜像对换。设图像的大小为 $M \times N$，水平镜像可按式（6-36）计算：

$$\begin{cases} i' = i \\ j' = N - j + 1 \end{cases} \tag{6-36}$$

式中，(i, j) 为原图像 $F(i, j)$ 中像素点的坐标；(i', j') 为对应像素点 (i, j) 水平镜像变换后图像 $H(i', j')$ 中的坐标。设原图像的矩阵为

$$\boldsymbol{F} = \begin{pmatrix} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} \\ f_{51} & f_{52} & f_{53} & f_{54} & f_{55} \end{pmatrix} \tag{6-37}$$

经过水平镜像的图像，行的排列顺序保持不变，将原来的列排列 $j = 1,2,3,4,5$ 转换成 $j' = 5,4,3,2,1$，即

$$\boldsymbol{F} = \begin{pmatrix} f_{15} & f_{14} & f_{13} & f_{12} & f_{11} \\ f_{25} & f_{24} & f_{23} & f_{22} & f_{21} \\ f_{35} & f_{34} & f_{33} & f_{32} & f_{31} \\ f_{45} & f_{44} & f_{43} & f_{42} & f_{41} \\ f_{55} & f_{54} & f_{53} & f_{52} & f_{51} \end{pmatrix} \tag{6-38}$$

2. 图像的垂直镜像

图像的垂直镜像操作是将图像上半部分和下半部分以图像水平中轴线为中心进行镜像对换。设图像的大小为 $M \times N$，垂直镜像可按式（6-39）计算：

$$\begin{cases} i' = M - i + 1 \\ j' = j \end{cases} \tag{6-39}$$

式中，(i, j) 为原图像 $F(i, j)$ 中像素点的坐标；(i', j') 为对应像素点 (i, j) 垂直镜像变换后图像 $H(i', j')$ 中的坐标。

设原始图像的矩阵见式（6-37），经过垂直镜像的图像，列的排列顺序保持不变，将原来行的排列 $i = 1,2,3,4,5$ 转换成 $i' = 5,4,3,2,1$，即

$$H = \begin{pmatrix} f_{51} & f_{52} & f_{53} & f_{54} & f_{55} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} \\ f_{11} & f_{12} & f_{13} & f_{14} & f_{15} \end{pmatrix} \tag{6-40}$$

3. 图像的对角镜像

图像的对角镜像操作是将图像以图像水平中轴线和垂直中轴线的交点为中心进行镜像对换。相当于将图像先后进行水平镜像和垂直镜像。设图像的大小为 $M \times N$，对角镜像可按式（6-41）计算：

$$\begin{cases} i' = M - i + 1 \\ j' = N - j - 1 \end{cases} \tag{6-41}$$

式中，(i, j) 为原图像 $F(i, j)$ 中像素点的坐标；(i', j') 为对应像素点 (i, j) 对角变换后图像 $H(i', j')$ 中的坐标。设原图像的矩阵见式（6-37），经过对角镜像的图像，将原来的行排列 $i = 1,2,3,4,5$ 转换成 i=5,4,3,2,1，将原来的列排列 $j = 1,2,3,4,5$ 转换成 $j' = 5,4,3,2,1$，即

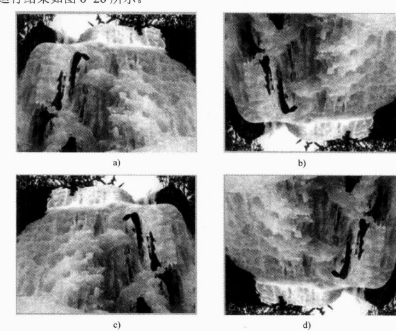
$$H = \begin{pmatrix} f_{55} & f_{54} & f_{53} & f_{52} & f_{51} \\ f_{45} & f_{44} & f_{43} & f_{42} & f_{41} \\ f_{35} & f_{34} & f_{33} & f_{32} & f_{31} \\ f_{25} & f_{24} & f_{23} & f_{22} & f_{21} \\ f_{15} & f_{14} & f_{13} & f_{12} & f_{11} \end{pmatrix} \tag{6-42}$$

4. 图像镜像变换的 MATLAB 实现

垂直、水平和对角镜像的程序如下：其中，I1 为原始图像；I2 为垂直镜像；I3 为水平镜像；I4 为对角镜像。

```
I1=imread('F:\image\fj.jpg');
I1=double(I1);
figure(1),
imshow(uint8(I1));
H=size(I1);
figure(2),
I2(1:H(1),1:H(2),1:H(3))=I1(H(1):-1:1,1:H(2),1:H(3)); %垂直镜像
imshow(uint8(I2));
figure(3),
I3(1:H(1),1:H(2),1:H(3))=I1(1:H(1),H(2):-1:1,1:H(3)); %水平镜像
imshow(uint8(I3));
figure(4),
I4(1:H(1),1:H(2),1:H(3))=I1(H(1):-1:1,H(2):-1:1,1:H(3)); %对角镜像
imshow(uint8(I4));
```

程序运行结果如图 6-26 所示。



a) b) c) d)

图 6-26 图像镜像变换

a) 原始图像 b) 垂直镜像 c) 水平镜像 d) 对角镜像

6.5.5 图像剪切变换

1. 图像剪切变换

当只需要处理图像中的一部分时，或者需要将某一部分取出，这样就要对图像进行剪切。MATLAB 7.0 图像处理工具箱提供了函数 imcrop()用于剪切图像中的一个矩形图，用户可以通过参数指定这个矩形顶点的坐标，也可以用鼠标指针选取这个矩阵。

函数 imcrop()的语法格式为

```
I2=imrcop(I)
X2= imrcop(X, map)
RGB2= imrcop(TGB)
I2= imrcop(I, rect)
X2= imrcop(X, map, rect)
RGB2= imrcop(RGB, rect)
[…]= imrcop(x , y, …)
[A, rect]= imrcop(…)
[x, y, rect]= imrcop(…)
```

其中，I2=imrcop(I)、X2= imrcop(X, map)和 RGB2= imrcop(TGB)为交互式地对灰度图像、索引色图像和真彩色图像进行剪切。I2= imrcop(I, rect)、X2= imrcop(X, map, rect)和 RGB2= imrcop(RGB, rect)按指定的矩形框 rect 剪切图像，rect 是一个四元向量[xmin ymin

width height]，四个分量分别表示矩形的左上角的坐标、宽度和高度。[…]= imrcop(x , y, …) 在指定坐标系(x, y)中剪切图像。[A，rect]= imrcop(…)和[x, y, rect]= imrcop(…)在用户手动选取剪切图像的同时返回剪切框的参数 rect。

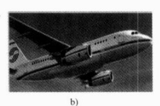
2. 图像剪切的 MATLAB 实现

下面的例子将从一幅图像中剪切一块子图像，坐标为(80, 70)～(220, 178)，结果如图 6-27 所示。

```
I1=imread('F:\image\jj.jpg');
I2=imcrop(I1,[75 68 220 100]);
imview(I1),imview(I2);
```



a)



b)

图 6-27　图像剪切变换

a) 原始图像　b) 剪切后的图像

6.5.6 图像复合变换

图像的复合变换是指对给定的图像进行两次或两次以上的平移、镜像、比例、旋转等基本变换的多次变换，又称为级联变换。由于引入齐次坐标后图像的基本变换采用了统一的矩阵表示形式，根据矩阵理论可知，对给定图像按顺序连续进行多次基本图像变换，其变换的矩阵仍然可以用 3×3 矩阵表示。复合变换的矩阵等于基本变换的矩阵按变换顺序依次相乘。

若对图像依次进行了 n 次平移、镜像、比例、旋转等基本变换，其变换矩阵分别为 $\boldsymbol{T}_1,\boldsymbol{T}_2,\cdots,\boldsymbol{T}_n$，则 n 次变换之后的复合变换矩阵 $\boldsymbol{T}$ 可以表示为

$$\boldsymbol{T}=\boldsymbol{T}_1\boldsymbol{T}_2\cdots\boldsymbol{T}_{n-1}\boldsymbol{T}_n \tag{6-43}$$

根据复合变换的组合类型，复合变换可以分为如下两类。

1. 同类型复合变换

复合变换由同一种基本变换组成，即相同的基本变换连续进行多次，如复合平移、复合比例缩放、复合旋转等。

2. 不同类型的复合变换

复合变换由不同类型的基本变换组成，即不同的基本变换连续进行多次，如图像的转置、绕任意点的比例缩放、绕任意点的旋转等。

现对同类型的复合变换讨论如下。

（1）复合平移

以包含两次基本平移的复合平移为例，指将图像先平移到新的位置 $A_1(x_1, y_1)$ 后，再将图像平移到位置 $A_2(x_2, y_2)$，复合平移矩阵为

$$\boldsymbol{T} = \boldsymbol{T}_1\boldsymbol{T}_2 \begin{pmatrix} 1 & 0 & \Delta x_1 \\ 0 & 1 & \Delta y_1 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & \Delta x_2 \\ 0 & 1 & \Delta y_2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x_1 + \Delta x_2 \\ 0 & 1 & \Delta y_1 + \Delta y_2 \\ 0 & 0 & 1 \end{pmatrix} \tag{6-44}$$

观察复合平移矩阵 $\boldsymbol{T}$ 可以发现，两次平移之后，平移的距离等于两次平移距离之和。因此，采用齐次坐标表示几何变换，不仅形式简洁，而且具有明显的几何意义。两次以上的复合平移变换可由此类推。

（2）复合比例

同样，以包含两次比例缩放变换的复合缩放变换为例，复合比例矩阵为

$$\boldsymbol{T} = \boldsymbol{T}_1\boldsymbol{T}_2 \begin{pmatrix} a_1 & 0 & 0 \\ 0 & b_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} a_2 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a_1a_2 & 0 & 0 \\ 0 & b_1b_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6-45}$$

对图像连续进行多次比例缩放变换，最后的复合比例矩阵，只需要对两次变换的比例常数进行乘积运算即可。

（3）复合旋转

类似地，对图像连续进行多次旋转变换，最后合成的旋转变换矩阵等于各次旋转角度之和。以包含两次旋转变换的复合旋转变换为例，复合旋转矩阵如下：

$$\boldsymbol{T} = \boldsymbol{T}_1\boldsymbol{T}_2 \begin{pmatrix} \cos\beta_1 & \sin\beta_1 & 0 \\ -\sin\beta_1 & \cos\beta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos\beta_2 & \sin\beta_2 & 0 \\ -\sin\beta_2 & \cos\beta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos(\beta_1+\beta_2) & \sin(\beta_1+\beta_2) & 0 \\ -\sin(\beta_1+\beta_2) & \cos(\beta_1+\beta_2) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6-46}$$

以上均为相对于原点（图像中心）进行比例、旋转等复合变换，如果要相对其他参考点进行以上变换，则要先进行平移，然后再进行其他基本变换，最后形成图像的复合变换。不同的复合变换，所包含的基本变换的数量和次序各不相同，但是无论其变换过程多么复杂，都可以分解成若干基本变换组成，都可以采用齐次坐标表示，且图像复合变换矩阵由一系列基本变换矩阵依次相乘而得到。

6.5.7 透视投影

在光线的照射下，三维物体可以在二维平面上形成投影，这种将三维物体或对象转变为二维图形表示的过程称为投影变换，投影中心称为视点。根据视点与投影平面之间距离的不同，投影可分为平行投影和透视投影。平行投影的视点与投影平面之间的距离为无穷大，若

该距离是有限的则称为透视投影。

透视投影即透视变换，透视投影的距离决定着透视投影的透视缩小效应，三维物体或对象透视投影的大小与物体到视点的距离成反比。例如，平行于投影面且长度相等的两段直线，离投影中心距离较近的线段，其透视投影长，离投影中心远的线段，其透视投影短。这种效应所产生的视觉效果与照相机系统、人的视觉系统十分相似。与平行投影相比，透视投影的深度感更强，看上去更真实，但透视投影不能真实地反映物体的精确尺寸和形状。

透视投影时，平行于投影面的平行线的投影依然平行，而不平行于投影面的平行线的投影则会聚集到一个点，这个点称为灭点（Vanishing Point）。灭点可以看做是无限远处的一点在投影面上的投影。透视投影的灭点可以有无限多个，不同方向的平行线在投影面上就可以形成不同的灭点，其中坐标轴方向的平行线在投影面上所形成的灭点称为主灭点。对于三维空间（x、y、z 3 个坐标轴）主灭点最多有 3 个。透视投影一般根据主灭点的数量（即按投影面与坐标轴的夹角）进行分类，因此可分为一点透视、二点透视和三点透视，如图 6-28 所示。

以一点透视为例，如图 6-28a 所示，它仅有一个主灭点，即投影面与一个坐标轴正交，与另外两个坐标轴平行。进行一点透视投影变换时，应考虑好投影的布局，以避免三维物体或对象的平面域积聚成直线或直线积聚成点而影响直观性。主要考虑如下几点：

1）三维物体或对象与画面（投影面）的相对位置。

2）视距，即视点与画面的距离。

3）视点的高度。

如图 6-29 所示，设视点在坐标原点，z 坐标轴方向与观察方向重合一致，点 $P(x,y,z)$ 为三维物体或对象上的一点，经一点透视变换后在投影面（观察平面）UOV 上的对应点为 $P'(x',y',z')$，投影面与视点的距离为 d，并与 z 轴垂直，z 轴过投影面窗口的中心，窗口是边长为 $2a$ 的正方形，根据几何学知识可得：

$$\begin{cases} \dfrac{x'}{x} = \dfrac{y'}{y} = \dfrac{z'}{z} = k \\ z' = d \end{cases} \tag{6-47}$$

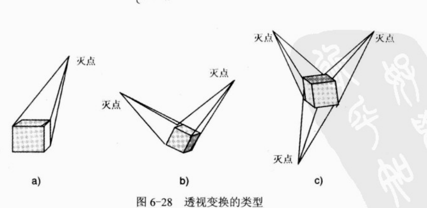


图 6-28　透视变换的类型

a) 一点透视　b) 二点透视　c) 三点透视

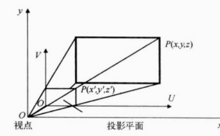


图 6-29 一点透视变换原理

利用齐次坐标，与二维几何变换类似，将该过程写成变换矩阵形式为

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{6-48}$$

这就是图像的一点透视变换，即

$$\begin{pmatrix} x' & y' & z' & 1 \end{pmatrix} = \begin{pmatrix} \frac{x}{z}d & \frac{y}{z}d & d & 1 \end{pmatrix} \tag{6-49}$$

实际上，一般情况下，一点透视变换矩阵也可以用一个 4×4 的矩阵表示。

6.5.8 平行投影

平行投影不具有缩小性，能精确反映物体的实际尺寸。平行线的平行投影仍是平行线。平行投影可根据投影方向与投影面的夹角分成正投影和斜投影两种。当投影方向与投影面的夹角为 90°时，得到的投影为正投影，否则为斜投影。是平行投影中的两类投影模式的原理示意图如图 6-30a、b 所示。

1．正投影

如图 6-30c、d 所示，依据投影平面的法矢量的方向，正投影又分为三视图和正轴侧两种模式。

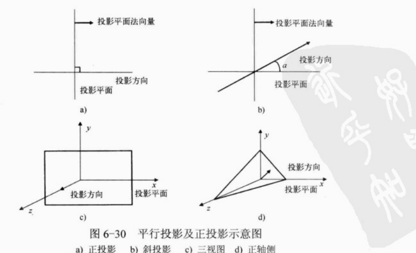


图 6-30 平行投影及正投影示意图
a) 正投影 b) 斜投影 c) 三视图 d) 正轴侧

当投影平面与某一坐标轴垂直时，得到的投影称为三视图。三视图分为主视图、侧视图和俯视图，对应的投影平面分别与 z 、 x 、 y 轴垂直。图 6-31 是一个物体（房屋）的三视图，从图中可以看到，从三个不同的方向得到的同一个物体的形状是不一样的。三视图常用于工程制图，因为在工程制图上可以测量距离和角度。但一个方向上的视图只反映物体的一个侧面，只有将三个方向上的视图结合起来，才能综合出物体的空间结构和形状。

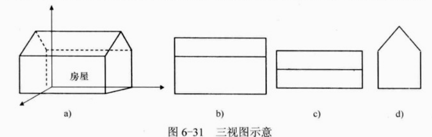


图 6-31 三视图示意

a) 三视图 b) 主视图 c) 俯视图 d) 侧视图

将三视图投影变换用齐次坐标来表示。设点 P 在三维空间的坐标为 (x, y, z) ，其投影点 P' 在三维空间上的坐标为 $(x_{p'}, y_{p'}, z_{p'})$ ，则在齐次坐标下，主视图、侧视图和俯视图的变换见式（6-50）～（6-52）。

$$\begin{pmatrix} x_{p'} & y_{p'} & z_{p'} & 1 \end{pmatrix} = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \boldsymbol{T}_{\text{front}} \qquad \boldsymbol{T}_{\text{front}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6-50}$$

$$\begin{pmatrix} x_{p'} & y_{p'} & z_{p'} & 1 \end{pmatrix} = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \boldsymbol{T}_{\text{side}} \qquad \boldsymbol{T}_{\text{side}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6-51}$$

$$\begin{pmatrix} x_{p'} & y_{p'} & z_{p'} & 1 \end{pmatrix} = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \boldsymbol{T}_{\text{top}} \qquad \boldsymbol{T}_{\text{top}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6-52}$$

非三视图的平行投影为正轴侧，如图 6-32 所示。正轴侧有等轴侧、正二侧和正三侧三种。当投影面与三个坐标轴之间的夹角都相等时为等轴侧；当投影面与两个坐标轴之间的夹角相等时为正二侧；当投影面与三个坐标轴之间的夹角都不相等时为正三侧。如图 6-33 所示，正轴侧可以通过几步分解获得。如图 6-33a 所示，投影面分别与三个坐标交于 A、B 和 C，投影方向与投影面垂直。首先，把物体及投影面绕 y 轴顺时针旋转 θ 角，得图 6-33b，再绕 x 轴逆时针旋转 φ 角，得图 6-33c。换句话说，就是将正轴侧分为两个绕坐标轴旋转的处理来完成。这样，用齐次坐标来描述得：

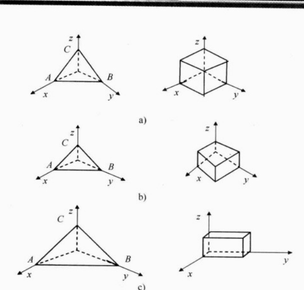


图 6-32 正轴侧正方体侧投影

a) 等轴侧 b) 正二侧 c) 正三侧

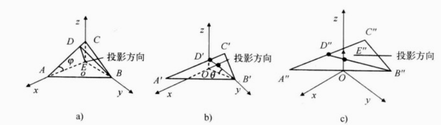


图 6-33 正轴侧投影变换分解示意图

a) 正轴侧投影平面 b) 投影面绕 y 轴顺时针旋 θ 角 c)投影面绕 x 轴顺时针旋转 φ 角

$$\begin{pmatrix} x_{p'} & y_{p'} & z_{p'} & 1 \end{pmatrix} = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \boldsymbol{T} \tag{6-53}$$

做绕 y 轴及 x 轴的旋转：

$$\boldsymbol{T} = \boldsymbol{T}_x \boldsymbol{T}_y = \begin{pmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\varphi & -\sin\varphi\sin\theta & \sin\varphi\cos\varphi & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ -\sin\varphi & -\cos\varphi\sin\theta & \cos\varphi\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6-54}$$

在 z 轴方向上做正投影，就得到正轴侧的投影变换矩阵为

$$T=\begin{pmatrix}\cos\varphi & -\sin\varphi\sin\theta & 0 & 0\\ 0 & \cos\theta & 0 & 0\\ -\sin\varphi & -\cos\varphi\sin\theta & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{6-55}$$

等轴侧的条件是投影面与三个坐标轴方向的夹角都相等，于是得到等轴侧的投影变换矩阵为

$$T_{iso}=\begin{pmatrix}\sqrt{2}/2 & -\sqrt{6}/6 & 0 & 0\\ 0 & \sqrt{6}/3 & 0 & 0\\ -\sqrt{2}/2 & -\sqrt{6}/6 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{6-56}$$

正二侧的条件是投影与某两个坐标轴之间的夹角相等。因此，正二侧的投影变换矩阵为

$$T_{\dim}=\begin{pmatrix}\sqrt{2}/2 & -\sqrt{2}/2\sin\theta & 0 & 0\\ 0 & \cos\theta & 0 & 0\\ -\sqrt{2}/2 & -\sqrt{2}/2\sin\theta & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{6-57}$$

2. 斜投影

斜投影是第二类平行投影，与正投影的区别在于投影方向与投影面不垂直。斜投影将正投影的三视图和正轴侧的特性结合起来，既能像三视图那样在主平面上进行距离和角度测量，又能像正轴侧那样同时反映物体的多个面，具有立体效果。通常选择投影面垂直于某个主轴，这样对平行于投影面的物体表面可进行距离和角度的侧量，而对物体的其他面，可沿该主轴侧量距离。

如图 6-34 所示，常用的两种斜投影是斜等侧和斜二侧。当投影方向与投影平面成 45°角时，得到的是斜等侧（见图 6-34a）。这时，和投影平面垂直的任何直线段，其投影长度不变，即图 6-34a 中 $AB = A'B'$。当投影方向与投影平面成 arctan 2 的角度时，得到的是斜二侧（图 6-34b）。这时，和投影平面垂直的任何直线，其投影长度为原来的一半，即图 6-34b 中 $AB = 2A'B'$。

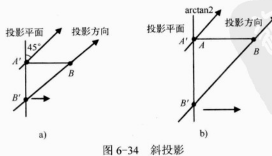


图 6-34 斜投影

a) 斜等侧 b) 斜二侧

有了斜投影的概念，下面来看斜投影变换。设投影平面为 xy 平面，投影方向与投影平

面的夹角为 α，投影线和 z 轴所组成的平面与 xz 面的两面角为 β。如图 6-35 所示，点 $P'(x_p^{(0)}, y_p^{(0)}, 0)$ 是点 $P(0,0,z)$ 在投影平面上的斜投影，设其大小为 k，于是有：

$$x_p^{(0)} = k\cos\beta \quad y_p^{(0)} = k\sin\beta \quad k = z\tan\alpha \tag{6-58}$$

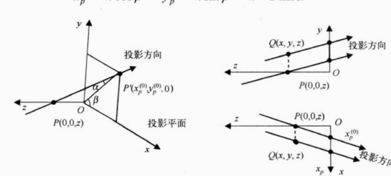


图 6-35 斜投影变换

对于空间任意一点 $Q(x,y,z)$，在投影平面上的斜投影 $O'(x,y,0)$ 的坐标可以直接得出，即

$$\begin{cases} x_p = x_p^{(0)} + x = z\tan\alpha\cos\beta + x \\ y_p = y_p^{(0)} + y = z\tan\alpha\sin\beta + y \end{cases} \tag{6-59}$$

斜投影变换矩阵为

$$\boldsymbol{T}_{\text{obl}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \tan\alpha\cos\beta & \tan\alpha\sin\beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6-60}$$

其中，斜等侧的投影方向与投影平面的夹角 $\alpha = 45°$，$\tan\alpha = 1$，所以变换矩阵为

$$\boldsymbol{T}_{\text{obliso}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \cos\beta & \sin\beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6-61}$$

对于斜二侧，投影方向与投影平面的夹角 $\alpha = \arctan\frac{1}{2}$，$\tan\alpha = 1/2 = 0.5$，所以变换矩阵为

$$\boldsymbol{T}_{\text{obldim}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5\cos\beta & 0.5\sin\beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6-62}$$

图 6-36 给出了一个单位立方体在 xy 平面上的几种斜投影。那些倾斜线是与 xy 平面垂直的立方体棱边的投影，它们与坐轴轴 x 的夹角，就是图 6-36 中的投影倾角 β，β 一般取 45° 和 30°。可以看到，同一个物体以不同的投影方式进行投影后，所得到的图形是不相同的。

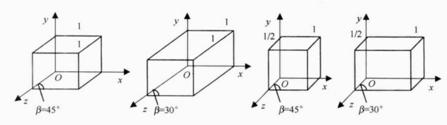


图 6-36　单位立方体的斜投影

6.6 灰度级插值

在进行图像的比例缩放、旋转及复合变换等，原始图像的像素坐标 (x,y) 为整数，而变换后目标图像的位置坐标并非整数，反过来也是如此，因此，在进行图像的几何变换时，除了要进行几何变换运算之外，还需要进行灰度级插值处理。常用的灰度级插值方法有三种：最近邻插值法、双线性插值法和三次内插值法。

6.6.1 最近邻插值法

最近邻法插值是一种简单的插值方法，如图 6-37 所示，它是通过计算与点 $P(x_0,y_0)$ 临近的四个点，并将与点 $P(x_0,y_0)$ 最近的整数坐标点 (x,y) 的灰度值取为 $P(x_0,y_0)$ 点灰度近似值。在 $P(x_0,y_0)$ 点各相邻像素间灰度变化较小时，这种方法是一种简单快速的方法，但当 $P(x_0,y_0)$ 点相邻像素间灰度值差异很大时，这种灰度估值方法会产生较大的误差，甚至可能影响图像质量。

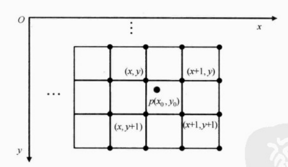


图 6-37　最近邻插值法插值示意图

6.6.2 双线性插值法

双线性插值法是对最近邻法的一种改进，即用线性内插方法，根据点 $P(x_0,y_0)$ 的四个相邻点的灰度值，通过两次插值计算出灰度值 $f(x_0,y_0)$，如图 6-38 所示。

具体计算情况如下：

1）计算 α 和 β。

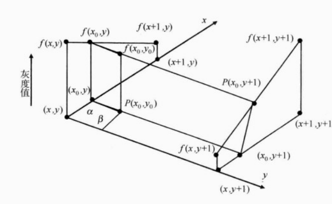


图 6-38 双线性插值法



$$\begin{cases}\alpha = x_0 - x \\ \beta = y_0 - y\end{cases} \tag{6-63}$$

2）先根据 $f(x,y), f(x+1,y)$ 插值求 $f(x_0,y)$。

$$f(x_0,y) = f(x,y) + a[f(x+1,y) - f(x,y)] \tag{6-64}$$

3）再根据 $f(x_0,y+1), f(x+1,y)$ 插值求 $f(x_0,y+1)$。

$$f(x_0,y+1) = f(x,y+1) + a[f(x+1,y+1) - f(x,y+1)] \tag{6-65}$$

4）最后根据 $f(x_0,y)$ 及 $f(x_0,y+1)$ 插值求 $f(x_0,y_0)$。

$$\begin{aligned} f(x_0,y_0) &= f(x_0,y) + \beta[f(x_0,y+1) - f(x_0,y)] \\ &= (1-\alpha)(1-\beta)f(x,y) + \alpha(1-\beta)f(x+1,y) + \\ &\quad (1-\alpha)\beta f(x,y+1) + \beta\alpha f(x+1,y+1) \\ &= f(x,y) + \alpha[f(x+1,y) - f(x,y)] + \beta[f(x,y+1) - f(x,y)] + \\ &\quad \beta\alpha[f(x+1,y+1) + f(x,y) - f(x,y+1) - f(x+1,y)] \end{aligned} \tag{6-66}$$

式中， $x = [x_0]; y = [y_0]$。

由于双线性插值法已经考虑到了点 $P(x_0,y_0)$ 的直接邻点对它的影响，因此一般可以得到令人满意的插值效果。但这种方法具有低通滤波性质（后面将介绍），使高频分量受到损失，使图像细节退化而变得轮廓模糊。在某些应用中，双线性插值的斜率不连续还可能会产生一些不期望的结果。

6.6.3 三次内插值法

为了得到更精确的 $P(x_0,y_0)$ 点的灰度值，在更高程度上保证几何变换后的图像质量，实现更精确的灰度插值效果，可采用三次内插值法等更高阶插值法，如三次样条函数、Legendre 中心函数和 $\sin(\pi x)/(\pi x)$ 函数等，这时既要考虑 $P(x_0,y_0)$ 点的直接邻点对它的影响，还应考虑到该点周围 16 个邻点的灰度值对它的影响（见图 6-37）。

根据连续信号采样定理可知，若对采样值用插值函数 $s(x) = \sin(\pi x)/(\pi x)$ 进行插值，当采样频率不低于信号谱最高频率的两倍时可以准确地恢复原信号，并可准确地得到采样点间任

意点的值。插值函数 sin(πx)/(πx)的特性如图 6-39 所示。

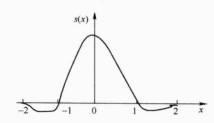


图 6-39 插值函数 sin(πx)/(πx)的特性

$s(x)=\sin(\pi x)/(\pi x)$ 可以采用以下三次多项式近似。

$$s(x)\begin{cases}1-2|x|^2+|x|^3 & |x|>1\\ 4-8|x|+5|x|^2-|x|^3 & 2>|x|\geqslant 1\\ 0 & |x|\geqslant 2\end{cases} \tag{6-67}$$

采用插值函数 $\sin(\pi x)/(\pi x)$，可按下述步骤插值算出 $f(x_0,y_0)$：

1）计算 $s(1+\alpha)$、$s(\alpha)$、$s(1-\alpha)$、$s(2-\alpha)$ 以及 $s(1+\beta)$、$s(\beta)$、$s(1-\beta)$、$s(2-\beta)$。

2）根据 $f(x-1,y), f(x,y), f(x+1,y), f(x+2,y)$ 计算 $f(x_0,y)$。

$$f(x_0,y)=s(1+\alpha)f(x-1,y)+s(\alpha)f(x,y)+s(1-\alpha)f(x+1,y)+s(2-\alpha)f(x+2,y) \tag{6-68}$$

3）按步骤 2）求 $f(x_0,y-1), f(x_0,y+1), f(x_0,y+2)$。

4）根据 $f(x_0,y-1), f(x_0,y), f(x_0,y+1), f(x_0,y+2)$ 计算 $f(x_0,y_0)$。

$$f(x_0,y_0)=s(1+\beta)f(x_0,y-1)+s(\beta)f(x_0,y)+s(1-\beta)f(x_0,y+1)+s(2-\beta)f(x_0,y+2) \tag{6-69}$$

上式计算过程可用矩阵表示为

$$\boldsymbol{A}=\begin{pmatrix}s(1+\alpha) & s(\alpha) & s(1-\alpha) & s(2-\alpha)\end{pmatrix}$$

$$\boldsymbol{B}=\begin{pmatrix}f(x-1,y-1) & f(x-1,y) & f(x-1,y+1) & f(x-1,y+2)\\ f(x,y-1) & f(x,y) & f(x,y+1) & f(x,y+2)\\ f(x+1,y-1) & f(x+1,y) & f(x+1,y+1) & f(x+1,y+2)\\ f(x+2,y-1) & f(x+2,y) & f(x+2,y+1) & f(x+2,y+2)\end{pmatrix}$$

$$\boldsymbol{C}=\begin{pmatrix}s(1+\beta) & s(\beta) & s(1-\beta) & s(2-\beta)\end{pmatrix}^{\mathrm{T}}$$

6.6.4 灰度级插值法的 MATLAB 实现

如图 6-40 所示，图中分别采用最近邻插值法、双线性插值法和三次内插值法得到的图



像放大结果。其实现代码如下：

```
I=imread('lena.bmp');
figure,imshow(I)
X1=imresize(I,1); %最近邻插值法
X2=imresize(I,1,'bilinear'); %双线性插值法
X3=imresize(I,1,'bicubic') %三次内插值法
figure,imshow(X1);
figure,imshow(X2);
figure,imshow(X3)
```

从以上三种插值方法对图像放大的结果可以看出，采用最近邻插值法的图像中明显可以看块状效应，但对于质量要求不高的情况下，图像效果还可以接受，双线性插值法和三次内插值法的结果则没有块状，但双线性插值法有些模糊，三次内插值法的效果是最好的。



a) b) c) d)

图 6-40 采用不同插值方法的图像

a) 原始图像 b) 最近邻插值法 c) 双线性插值法 d) 三次内插值法

习题

6-1 为什么点运算不会改变图像内像素的空间位置关系？

6-2 图像相加运算能消除图像的加性随机噪声吗？为什么？

6-3 什么是齐次坐标？在图像的几何变换中，它具有哪些优点？

6-4　几何运算在数字图像处理技术中具有哪些典型应用？

6-5　常用的几何运算有哪几种？

6-6　编写一个程序以实现如下功能：将一个灰度图像与该灰度图像经过平移后（边界全部填充为零）得到的图像，进行相减后再相乘，并显示和比较两种操作带来的不同图像输出效果。

6-7　图像旋转变换对图像的质量有无影响？为什么？

6-8　什么是图像的复合变换？复合变换可以分为几种情况？

6-9　设原图像为

59	60	58	57
61	59	59	57
62	59	60	58
59	61	60	56

请用最近邻插值法将该图像放大为 16×16 大小的图像。

第7章 图 像 增 强



经图像信息输入系统获取的原图像中通常都含有各种各样的噪声和畸变，大大影响了图像的质量。因此，在对图像进行分析之前，必须先对图像质量进行改善。通常，采用图像增强的方法对图像质量进行改善。图像增强不会考虑引起图像质量下降的原因，而是将图像中感兴趣的特征有选择地突出，并衰减不需要的特征。图像增强的目的是为了改善图像的视觉效果，提高图像的清晰度和工艺的适应性，以及便于人与计算机的分析主处理，以满足图像复制或再现的要求。

图像增强的方法分为空域法和频域法两类，空域法主要是对图像中的各个像素点进行操作；而频域法是在图像的某个变换域内对整个图像进行操作，并修改变换后的系数，如傅里叶变换、DCT 变换等的系数，然后再进行反变换，便可得到处理后的图像。

当然，在图像增强的过程中，总是以对某一部分信息的强调和另一部分信息的损失为代价的。因而总是在要求不降低图像质量的前提下，对图像进行处理来改善图像质量。但是，目前在图像增强方面还没有统一的质量评价标准，即缺乏从图像外观的角度进行主、客观评判的数学量度，所以图像增强手段还有待于更深入的研究。下面将对这一章展开学习。

7.1 灰度变换增强

灰度变换增强是根据某种目标条件，按一定变换关系逐点改变原图像中每一个像素点的灰度值的方法，即设原图像像素的灰度值为 $D=f(x,y)$，处理后图像像素的灰度值为 $D'=g(x,y)$，则灰度增强可表示为

$$f(x,y)=T[g(x,y)]$$
$$D'=T(D) \tag{7-1}$$

通过变换，达到对度增强的效果。当灰度变换关系 $D'=T(D)$ 确定后，则确定了一个具体的灰度增强方法。$D'=T(D)$ 通常是一个单值函数。

7.1.1 像素及其统计特性

1. 像素的选择

MATLAB 7.0 图像处理工具箱提供了两个指定像素信息的函数。

1）pixval( )函数：交互式显示像素的数值，也可以显示两个像素间的欧几里德距离。

2）impixel( )函数：返回被选择像素或像素集合的数据值，可通过输入参数定义像素坐标。以下命令行说明了 impixel( )函数的使用方法。

```
[C, R, P]=impixel(X, MAP)
```

其中，X 表示输入图像，MAP 为索引色图像的调色板（当图像为索引色图像时才有此参数），C 表示指定像素的颜色，R 和 P 表示像素的坐标。如果在输入图像参数后面给出两个指定的像素坐标的向量，则 impixel()函数将返回指定像素的灰度；如果调用 impixel()函数时没有指定参数，则系统将自动选择位于当前坐标轴中的图像。在交互式下，选择完毕后按〈Enter〉键，函数将返回被选像素的颜色数据。以下程序代码说明了该函数的操作情况。

```
imshow canoe.tif
vals=impixel
vals=
0.2902 0.2588 0.1922
0.5176    0      0
0.3882 0.3882 0.2902
```

显示图形效果如图 7-1 所示。

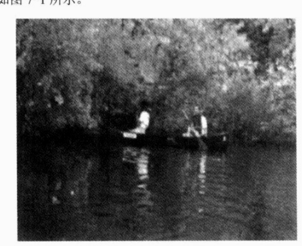


图 7-1　获取像素值示意图

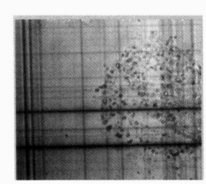
2．线段上像素灰度分布的计算和绘制

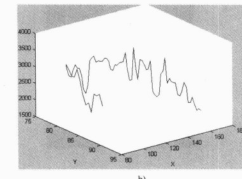
如果需要计算并绘制图像中一条或多条线段上的所有像素灰度值，MATLAB 7.0 图像处理工具箱提供的 improfile()函数可以实现该功能。调用该函数时，可以使用端点坐标作为输入参数来定义线段，也可以使用鼠标交互式地定义线段。非交互式 improfile()函数的调用格式如下：

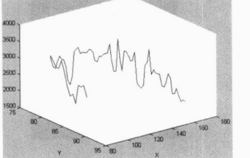
● C=improfile(I, Xi,Yi)

其中，I 为输入图像，Xi()和 Yi 是两个向量，用来指定线段的端点，输出为线段各点的灰度或颜色值。如果 improfile()函数不输入任何参数，那么当鼠标位于图像中时会变为十字形，可以通过鼠标来定义线段的端点。单击“返回”按钮后，improfile()函数将在一个新图形窗口中显示所得的线段灰度值的分布情况，如图 7-2 所示。

```
I=fitsread('solarspectra.fts');
imshow(I,[]);
improfile
```







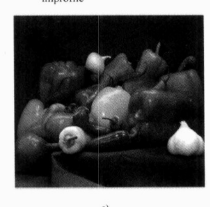
a) b)

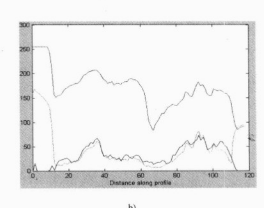
图 7-2 线段上的像素灰度分布图

a) 原始图像 b) 像素灰度分布曲线

图 7-3 所示是对 RGB 图像进行以下操作后显示的结果：

```
imshow peppers.png
improfile
```





a) b)

图 7-3 RGB 图像线段灰度分布图

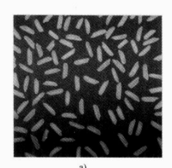
a) 原始图像 b) 像素灰度分布曲线

3. 图像等高线

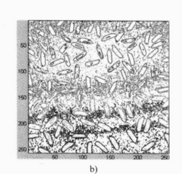
在 MATLAB 7.0 图像处理工具箱中还可以用 imcontour()函数来显示灰度图像的等高线。该函数与 MATLAB 7.0 中的 contour()函数类似，不同的就是 imcontour()函数可以自动进行坐标设置，使输出图像的方向和外观与图像吻合。以下程序代码示例说明了该函数的使用方法。

```
I=imread('rice.png');
imshow(I)
figure,imcontour(I)
```

执行程序代码后效果如图 7-4 所示。



a)



b)

图 7-4　图像与其等高线图

a) 原始图像　b) 等高线图像

4. 统计摘要

此外，还可以使用图像处理工具箱中的 mean2()函数、std2()函数和 corr2()函数，对图像的标准统计特性进行计算。

- mean2：计算矩阵元素的平均值
- std2：计算矩阵的标准偏差
- corr2：计算两个矩阵的相关系数

5. 区域属性度量

在 MATLAB 7.0 中可以用 regionprops()函数来计算图像的区域属性。这些属性包括测量指定图像区域的面积、质心、边框等。regionprops()函数调用方式如下：

```
BW=imread('text.png');
L=bwlabel(BW);
stats=regionprops(L,'all');
stats(23)
```

得到的图像属性统计结果为

```
ans =
        Area: 48
        Centroid: [121.3958 15.8750]
        BoundingBox: [118.5000 8.5000 6 14]
        SubarrayIdx: {[9 10 11 12 13 14 15 16 17 18 19 20 21 22]  [119 120 121 122 123 124]}
        MajorAxisLength: 15.5413
        MinorAxisLength: 5.1684
        Eccentricity: 0.9431
        Orientation: -87.3848
        ConvexHull: [10x2 double]
        ConvexImage: [14x6 logical]
        ConvexArea: 67
        Image: [14x6 logical]
        FilledImage: [14x6 logical]
        FilledArea: 48
        EulerNumber: 1
```



```
Extrema: [8x2 double]
EquivDiameter: 7.8176
Solidity: 0.7164
Extent: 0.5714
PixelIdxList: [48x1 double]
PixelList: [48x2 double]
Perimeter: 35.3137
```

7.1.2 直接灰度变换

在扫描过程中，由于扫描系统或者光电转换系统多方面的原因，常出现图像不均匀、对比度不足等弊端，使人眼在观看图像时视觉效果很差。灰度变换就是在图像采集系统中对图像像素进行修正，使整幅图像成像均匀。

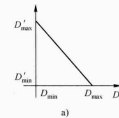
灰度变换可以分为 3 种：线性变换、分段线性变换和非线性变换。灰度变换可以使图像动态范围加大，图像对比度扩展、图像清晰、特征明显，是图像增强的重要手段。

1. 灰度线性变换

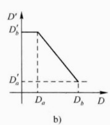
若 $D' = T(D)$ 是一个线性单值函数、则由它确定的灰度变换称为灰度线性变换，简称线性变换。

若图像灰度值 $D = f(x,y)$ 的可能值域为 $[D_{\min}, D_{\max}]$，如图 7-5 所示是一种 $D'-D$ 的曲线正斜率的线性变换，其表达式为

$$D' = D'_{\max} = \frac{D'_{\max} - D'_{\text{mim}}}{D_{\max} - D_{\text{mim}}}(D - D_{\min}) \tag{7-2}$$



a)



b)

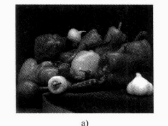
图 7-5　灰度线性变换

a) 灰度倒置　b) 局部斜率变换

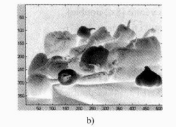
具体变换时，将图像中每个像素的灰度值根据变换曲线进行映射。

下面是灰度倒置线性变换的程序示例，其效果如图 7-6 所示。

```
clc;
clear all;
I=imread('peppers.png');
colormap;imshow(I);
%设置图像倒置参数
j=imadjust(I,[0 1],[1 0],1.5);
%显示倒置后的图像
figure;subimage(j)
```



a)



b)

图 7-6　灰度倒置结果

a) 原始图像　b) 灰度倒置后的图像

2. 分段线性变换

分段线性变换是将图像的值域分成多个值域并进行不同线性变换的一种算法。采用分段线性变换，可根据变换要求，压缩一部分灰度区间，扩展为另外一部分灰度区间。其变换表达式为

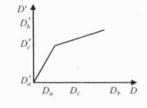
$$D' = \frac{D_c' - D_a'}{D_c - D_a}(D - D_a) + D_a' \qquad D \in [D_a, D_c] \tag{7-3}$$

$$D' = \frac{D_b' - D_c'}{D_b - D_c}(D - D_c) + D' \qquad D \in [D_c, D_b] \tag{7-4}$$

分段线性变换既可以用多段折线构成一个单位函数，又可逼近一条曲线。同时，经过这种变换以后，可使所关心的图像细节的灰度范围得以扩展，增强其对比度：同时，又使不关心的图像细节所处的灰度范围得以压缩，降低其对比度。值得注意的是，采用这种分段线性变换，变换前后整幅图像总的灰度范围是不变的，如图 7-7 所示。

3. 灰度的非线性变换

灰度的非线性变换简称非线性变换，是指由 $D' = T(D)$ 这样一个非线性单值函数所确定的灰度变换。这里主要讨论实际应用中经常使用的对数变换。对数变换常用来扩展低值灰度，压缩高值灰度，这样可以使低值灰度的图像细节更容易看清，从而达到图像增强的效果。对数变换曲线如图 7-8 所示，其表达式为



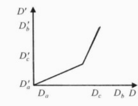


图 7-7　分段线性变换

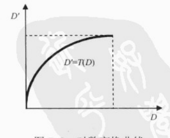


图 7-8　对数变换曲线

$$D' = C * \log(1 + |D|) \tag{7-5}$$

式中，C 为尺度比例常数。

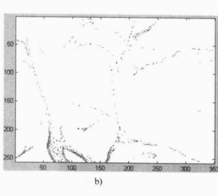
利用图 7-8 中的对数变换曲线，对图 7-9a 中的原始图像进行变换，可得到如图 7-9b 所

示的变换效果，其变换程序说明如下：

```
I=imread('trees.tif');colormap
imshow(I)
J=double(I);
J=45*log(J+1);
I=uint8(J);
figure,subimage(J)
```



a)



b)

图 7-9 对数变换前、后的图像效果图

a) 原始图像 b) 变换后的图像



7.1.3 直方图灰度变换

有多种方法可以实现图像灰度变换，其中最常用的就是直方图灰度变换的方法，下面将重点讨论直方图灰度变换方法。

图像的直方图是图像的重要统计特征，可以认为是图像灰度分布密度函数的近似。通常图像的灰度分布密度函数与像素所在的位置有关。设图像在点(x,y)处的灰度分布密度函数为$p(z;x,y)$，那么图像的灰度密度函数为

$$p(z)=\frac{1}{S}\iint_D p(z;x,y)\mathrm{d}x\mathrm{d}y \tag{7-6}$$

式中，D是图像的定义域；S是区域D的面积。一般来讲，要得到精确的图像灰度分布密度函数比较困难，所以实际中用图像的直方图来代替。直方图是一个离散函数，它表示数字图像每一灰度级与该灰度级出现频率的对应关系。设一幅数字图像的像素总数为N，有L个灰度级，具有第k个灰度级的灰度r_K的像素共有n_K个。则第k个灰度级出现的概率为

$$h_K=\frac{n_K}{N} \qquad k=0,1,\cdots,L-1 \tag{7-7}$$

在 MATLAB 7.0 图像处理工具箱中，直接提供了 imhist()函数来计算和显示图像的直方图，格式为

- imhist(I, n)：对灰度图像
- imhist(X, map)：对索引色图像

● [counts, x]=imhist(···)

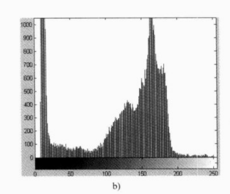
其中，I 代表灰度图像，n 为指定的灰度级数目，默认值为 256，counts 和 x 分别为返回直方图数据向量和相应的色彩值向量。

灰度直方图程序代码如下，操作效果如图 7-10 所示。

```
I=imread('cameraman.tif');
imshow(I,[40 255]);
figure,imhist(I)
```



a)



b)

图 7-10 灰度图像与直方图

a) 原始图像 b) 直方图

由图 7-10 可以看出，可以通过改变直方图的形状来达到增强图像对比度的效果，该方法以概率论为基础。实际上，改变直方图形状的常用方法就是直方图的均衡化。

在 MATLAB 7.0 中，图像处理工箱箱提供了对比度调整函数 imadjust()，用于调整灰度值或颜色图，其调用格式为

● J=imadjust(I, [low_in] high_in,[low_out high_out], γ)

将灰度图像 I 转化为图像 J，使值 low_in 到 high_in 与从 low_out 到 high_out 相匹配，大于 high_in 或小于 low_in 的值将被减去，即小于 low_in 的值与 low_out 相匹配，大于 high_in 的值与 high_out 相匹配。它们的默认值为[0，1]。γ 用来指定描述 I 和 J 值关系曲线的形状；γ <1 时，输入越亮，输出值越加强；γ >1 时，输入越亮，输出值越减弱；默认时 γ =1，表示线性变换。图 7-11 给出了 γ 值不同时对应的校正函数曲线。

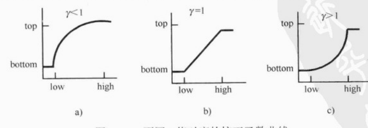


图 7-11 不同 γ 值对应的校正函数曲线

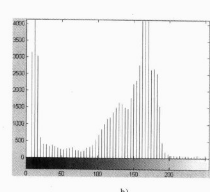
a) γ <1 时 b) γ =1 时 c) γ >1 时

下面将通过示例来说明 imadjust( )函数的用法。

在此仍然对图 7-10 所示的图像进行处理，程序代码如下，其处理结果如图 7-12 所示。对这两幅图进行对比，可以看出，经过变换后的图像覆盖了整个灰度范围，而在图像 7-10 中，灰度范围只在 40～255 之间。



a)



b)

图 7-12 均衡后的图像及其直方图

a) 变换后的图像 b) 变换后图像的直方图

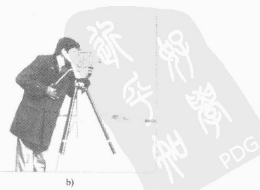
```
I=imread('cameraman.tif');
J=imadjust(I,[0.15 0.9],[0 1]);
imshow(J);
figure,imhist(I,64)
```

其中，imadjust( )函数的第二个向量[0.15, 0.9]指定需要映射的灰度值范围，第三个向量[0 1]指定希望映射到的灰度值范围，即灰度值 0.15 映射到输出图像中的 0，灰度值 0.9 映射到输出图像中的 1。

除了增强或减弱图像的对比度外，还可以对图像的灰度范围进行映射。以下程序代码示例就是用 imadjust( )函数将输入图像的灰度范围从[0 51]映射到[128 255]，输出图像如图 7-13 所示。



a)



b)

图 7-13 亮度调节前、后图像

a) 原始图像 b) 亮度调节后的图像

```
I=imread('cameraman.tif');
J=imadjust(I,[0 0.2],[0.5 1]);
imshow(I);
figure,imshow(J)
```

上述操作可以大大提高图像的亮度，也可使原始图像灰暗部分的动态变化范围大大增加，从而使细节更容易观看。

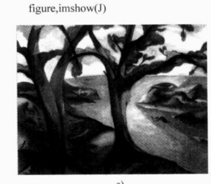
从上面的例子可以看出，用 imadjust( )函数必须按照下面的两个步骤进行：

1）观察图像的直方图，判断灰度范围。

2）将灰度范围转换为 0～1 之间的小数，使得灰度范围可以通过向量[low_in high_in]传递给函数。

此外，可选参数γ对图像修正的处理效果有很好的改善，如以下程序代码示例，仍假设图像灰度范围不变，取γ=0.5，对图 7-14a 进行变换，可得到如图 7-14b 所示的效果。

```
[X,map]=imread('trees.tif');
I=ind2gray(X,map);
J=imadjust(I,[],[],0.5);
figure,imshow(I)
figure,imshow(J)
```


a)

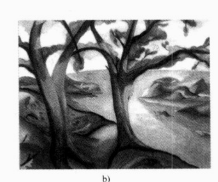

b)

图 7-14　γ 修正前、后的图像

a) 原始图像　b) 修正后的图像

7.1.4 直方图均衡化

直方图均衡化是一种使输出图像直方图近似服从均匀分布的变换算法，其计算步骤如下：

1）列出原始图像的灰度级 $f_j, j=0,1,\cdots,k,\cdots,L-1$，其中 L 是灰度级的个数。

2）统计各灰度级的像素数目 $n_j, j=0,1,\cdots,k,\cdots,L-1$。

3）计算原始图像直方图各灰度级的频度 $P_f(f_j)=\dfrac{n_j}{n}$，$j=0,1,\cdots,k,\cdots,L-1$，其中 n 为原始图像总的像素数目。

4）计算累计分布函数 $C(f)=\sum_{j=0}^{k}P_f(f_j)$，$j=0,1,\cdots,k,\cdots,L-1$。

5）应用以下公式计算映射后的输出图像的灰度级 g_i， $i=0,1,\cdots,k,\cdots,P-1$， P 为输出图像灰度级的个数：

$$g_i = \mathrm{INT}[(g_{\max} - g_{\min})C(f) + g_{\min} + 0.5]$$

式中，INT 为取整符号。

6）统计映射后各级灰度级的像素数目 n_i， $i=0,1,\cdots,k,\cdots,P-1$。

7）计算输出图像直方图 $P_g(g_i)=\frac{n_i}{n}$， $i=0,1,\cdots,k,\cdots,P-1$。

8）用 f_j 和 g_i 的映射关系修改原始图像的灰度级，从而获得直方图近似为均匀分布的输出图像。

以上 8 个步骤就是实现直方图均衡化的具体过程。MATLAB 7.0 的图像处理工具箱提供了图像直方图均衡化的具体函数 histeq( )，其具体调用格式为

- J=histeq(I, hgram)
- J=histeq(I, n)
- [J,T]=histeq(I,⋯)
- newmap=histeq(X,map,hgram)
- newmap=histeq(X, map)
- [newmap , T]=histeq(X,⋯)

其中，前面两行命令是对灰度图像而言，后三行命令是对索引色图像而言，具体用法可参照前面工具箱介绍。

以下程序代码示例说明了 histeq( )函数的用法，处理效果如图 7-15 所示。

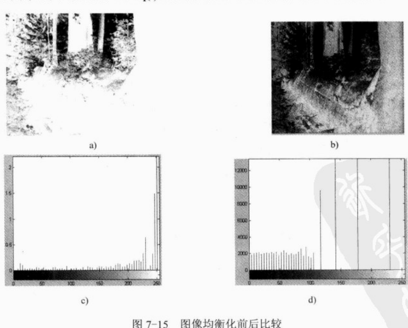


图 7-15 图像均衡化前后比较

a) 原始图像均衡前 b) 原始图像均衡后 c) 直方图均衡前 d) 直方图均衡后

```
I=imread('forest.tif');
J=histeq(I);
figure,imshow(I)
figure,imshow(J)
figure,imhist(I,64)
figure,imhist(J,64)
```

由图 7-15 可知，经过直方图均衡化后，可以看出图像的细节成分更加清楚了。同时，也可看出，在直方图调整之前，低灰度的比例很大，经过直方图调整后，各灰度等级的比例更加平衡。然而，由于直方图均衡没有考虑图像的内容，只是简单地将图像进行直方图均衡化，使图像看起来亮度过高。

7.1.5 对比度自适应直方图均衡化

在某些应用中，可能只需要对图像的某个部分进行均衡化，为此，可以用 adapthisteq() 函数替代 histeq()函数，实现对图像进行对比度自适应直方图均衡化。histeq()函数作用整幅图像时，可用 adapthisteq()函数对图像中的一个区域进行操作，称为切片，则切片的对比度将得到加强，这样，就可以使输出区域的直方图与指定的直方图相匹配。为了避免放大图像中噪声的对比度，可以设定 adapthisteq()函数的参数，限制噪声的对比度。

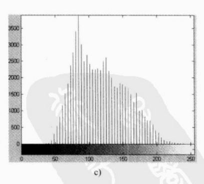
以下程序代码示例说明了如何利用 adapthisteq()函数来调整图像的对比度问题。如图 7-16a 所示的原始图像，其对比度比较低，大部分值集中在一起，经过直方图均衡化处理，可得到如图 7-16b 所示的效果，均衡化的直方图如图 7-16c 所示。

```
I=imread('pout.tif');
J=adapthisteq(I);
figure,imshow(I)
figure,imshow(J)
figure,imhist(J,64)
```



a)

b)



c)

图 7-16　自适应直方图均衡化前、后效果图

a) 原始图像　b) 处理后的图像　c) 处理后的直方图

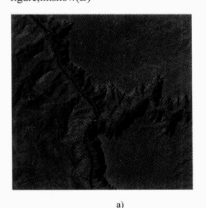
7.1.6 去相关拉伸

去相关拉伸可以增强图像相关区域的颜色，可以利用 MATLAB 7.0 图像处理工具箱中

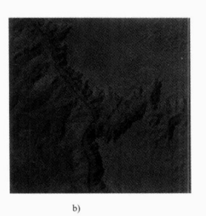
的 decorrstretch()函数实现去相关操作。

为了说明如何利用该函数实现去相关操作，这里以 imdemos 文件夹中的二进制图像为例进行说明。在本例中，图像有 7 个颜色段，但在这里只读取 3 个可见的颜色段。图 7-17 说明了原图像和处理后的效果。

```
A=multibandread('littlecoriver.lan',[512,512,7],'uint8=>uint8',128,'bil','ieee-le',{'Band','Direct',[3,2,1]});
B=decorrstretch(A);
figure,imshow(A);
figure,imshow(B)
```



a)

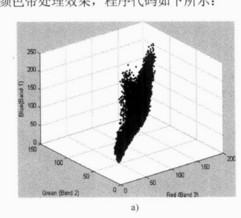


b)

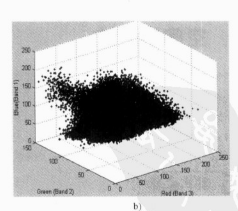
图 7-17　图像去相关拉伸前、后的效果图

a) 原始图像　b) 去相关拉伸后的图像

下面，将说明如何在一个颜色带内实现去相关和均衡化操作，图 7-18 说明了对图 7-17a 的颜色带处理效果，程序代码如下所示：



a)



b)

图 7-18　处理前、后颜色分散效果图

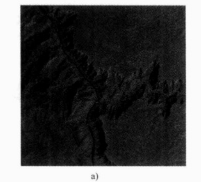
a) 处理前的颜色分散效果图　b) 处理后的颜色分散效果图

```
Ra=A(:,:,1);Ga=A(:,:,2);Ba=A(:,:,3);
figure,plot3(Ra(:),Ga(:),Ba(:),'.');grid('on')
```

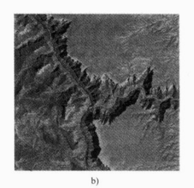
```
xlabel('Red (Band 3)');
ylabel('Green (Band 2)');
zlabel('Blue(Band 1)');
Rb=B(:,:,1);Gb=B(:,:,2);Bb=B(:,:,3);
figure,plot3(Rb(:),Gb(:),Bb(:),'.');grid('on');
xlabel('Red (Band 3)');
ylabel('Green (Band 2)');
zlabel('Blue(Band 1)');
```

当然，也可以在去相关拉伸后使用线性对比度拉伸，读者可以对比对比度拉伸前后的效果（见图 7-19）。

```
imshow(A);
C=decorrstretch(A,'Tol',0.01);
figure,imshow(C)
```



a)



b)

图 7-19 线性对比度拉伸后的效果图

a) 原始图像 b) 线性拉伸后的图像

7.2 空间域滤波

通常情况下，像素的领域比该像素要大，也就是说这个像素的领域中除了本身以外还包含了其他像素。在这种情况下，$g(x,y)$ 在 (x,y) 位置处的值不仅取决于其在该处的值，而且取决于以 (x,y) 为中心的领域内所有像素的值。

为在领域内实现图像增强操作，常可利用模板与图像进行卷积，每个模板实际上是一个二维数组，其中各个元素的取值确定了模板的功能，这种模板操作也称为空域滤波。

7.2.1 基本原理

根据其特点，空域滤波一般可分为线性滤波和非线性滤波两类。线性滤波器的设计常基于对傅里叶变换的分析。非线性空域滤波器则一般直接对领域进行操作。另外各种空域滤波器根据功能又主要分成平滑滤波器和锐化滤波器。平滑可用低通滤波来实现，平滑的目的可分为两类：一类是模糊，目的是在提取较大的目标前去除太小的细节或将目标内的小间断连接起来；另一类是消除噪声。锐化可用高通滤波来实现，锐化的目的是为了增强被模糊的细节。

结合这两种分类法，可将空间滤波增强方法分成 4 类：

- 线性平滑滤波器（低通）
- 非线性平滑滤波器（低通）
- 线性锐化滤波器（高通）
- 非线性锐化滤波器（高通）

空域滤波器的工作原理都可借助频域进行分析。它们的基本特点都是让图像在傅里叶空间的某个范围的分量受到抑制，而让其他分量不受影响，从而改变输出图像的频率分布，达到图像增强的目的。在图像增强中用到的空间滤波器主要有两类：

平滑（低通）滤波器：它能减弱或消除傅里叶空间的高频分量，但不影响低频分量。因为高频分量对应图像中的区域边缘等灰度值具有较大较快变化的部分，滤波器将这些分量滤去可使图像平滑。

锐化（高通）滤波器：它能减弱或消除傅里叶空间的低频分量，但不影响高频分量。因为低频分量对应图像中灰度值缓慢变化的区域，因而与图像的整体特性，如整体对比度和平均灰度值等有关，将这些分量滤去可使图像锐化。

空域滤波器都是利用模板卷积，主要步骤是：

1）将模板在图中漫游，并将模板中心与图中某个像素位置重合。

2）将模板上的系数与模板下对应的像素相乘。

3）将所有乘积相加。

4）将模板的输出响应赋给图中对应模板中心位置的像素。

下面重点介绍 MATLAB 中如何具体实现图像的平滑和锐化，以及此方法在工程中的应用实例。

7.2.2 平滑滤波

1. 线性平滑滤波

线性低通滤波器是最常用的线性平滑滤波器。实现这种滤波的方法也称为领域平均法。领域平均法是一种局部空间域处理的算法，这种方法的基本思想是用几个像素灰度的平均值来代替每个像素的灰度。假定有一幅 $N \times N$ 个像素的图像 $f(x,y)$，平滑处理后得到一幅图像 $g(x,y)$。$g(x,y)$ 由下式决定：

$$g(x,y)=\frac{1}{M}\sum_{(m\cdot n)\in S} f(m\cdot n) \tag{7-8}$$

式中，$x,y=0,1,2,\cdots,N-1$；S 是 (x,y) 点领域中点的坐标的集合，但其中不包括 (x,y) 点；M 是集合内坐标点的总数。式（7-8）说明，平滑后的图像 $g(x,y)$ 中的每个像素的灰度值均由：含在 (x,y) 的预定领域中的 $f(x,y)$ 的几个像素的灰度值的平均值来决定。一种常见的平滑算法是将原图像中一个像素的灰度值和它周围邻近 8 个像素的灰度值相加，然后求得的平均值（除以 9）作为新图像中该像素的灰度值。我们用如下方法来表示该操作：

$$\frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix} \tag{7-9}$$

中间的黑点表示该元素为中心元素，即该个元素是要进行处理的元素，同理可得 5×5 的模板，如下所示：

$$\frac{1}{25}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{7-10}$$

通常模板不允许移出边界，因此处理后的图像会比原图像小。对于边界上无法进行模板操作的点，我们的做法是复制原图像的灰度值，不再进行任何其他的处理。

另外，Wiener 滤波器也是经典的线性降噪滤波器。Wiener 滤波的思想是 20 世纪 40 年代提出来的，是一种在平稳条件下采用最小方均误差准则得出的最佳滤波准则，该方法就是寻找一个最佳的线性滤波器，使得方均误差最小。其实质是解维纳-霍夫（Wiener Hoof）方程。

Wiener 滤波器首先估计出像素的局部矩阵均值和方差：

$$\mu = \frac{1}{NM}\sum_{n1,n2\in\eta} a(n1,n2) \tag{7-11}$$

$$\sigma^2 = \frac{1}{NM}\sum_{n1,n2\in\eta} a^2(n1,n2) - \mu^2 \tag{7-12}$$

η 是图像中每个像素 $N \times M$ 的领域，利用 Wiener 滤波器估计出其灰度值：

$$b(n1,n2) = \mu + \frac{\sigma^2 - v^2}{\sigma^2}(a(n1,n2) - \mu) \tag{7-13}$$

式中，v^2 是整幅图像的方差。它根据图像的局部方差来调整滤波器的输出，当局部方差大时，滤波器的效果较弱，反之滤波器的效果较强，是一种自适应滤波器。

2．线性平滑滤波器的 MATLAB 应用

1）用 MATLAB 7.0 图像处理工具箱中提供的 imfilter()函数实现 5×5 领域平均运算。程序代码如下：

```
clear all;
I=imread('F:\image\child.bmp');
J=imnoise(I,'salt & pepper',0.02);
h=ones(5,5)/25;
I2=imfilter(J,h);
figure,imshow(J);
figure,imshow(I2)
```

采用领域平均法对图 7-20a 中的图像进行处理后的结果如图 7-20b 所示。可以看出经过领域平均处理后，图像的噪声得到了抑制，但图像变得相对模糊了。



a)



b)



图 7-20　5×5 领域平均运算得到的平滑图像

a) 有噪声的图像　b) 5×5 领域平均运算后的图像

2）Wiener 滤波实现降噪的过程，程序代码如下：

```
clear all;
I=imread('F:\image\lena.bmp');
figure,imshow(I);
K1=wiener2(I,[3,3]);   %3×3Wiener 滤波
K2=wiener2(I,[5,5]);   %5×5Wiener 滤波
K3=wiener2(I,[7,7]);   %7×7Wiener 滤波
figure,imshow(K1);
figure,imshow(K2);
figure,imshow(K3);
```

程序执行后效果如图 7-21 所示。



a)



b)



c)



d)

图 7-21　Wiener 滤波法实现降噪

a) 原始图像　b) 3×3Wiener 滤波　c) 5×5Wiener 滤波　d) 7×7Wiener 滤波

3．非线性平滑滤波器

中值滤波是一种去除噪声的非线性处理方法，是由 Turky 在 1971 年提出的。其基本原理是把数字图像或数字序列中一点的值用该点的一个领域中各点值的中值代替。中值的定义如下：一数组 $x_1,x_2,x_3,\cdots,x_n$，把 n 个数按值的大小顺序排列于下：$x_{i1}\leqslant x_{i2}\leqslant x_{i3}\leqslant\cdots\leqslant x_{in}$

$$y=\mathrm{med}\{x_1,x_2,x_3,\cdots,x_n\}=\begin{cases} x_{i\frac{(n+1)}{2}} & n\text{为奇数} \\ \dfrac{1}{2}\left[x_{i\frac{(n+1)}{2}}+x_{i(\frac{n}{2}+1)}\right] & n\text{为偶数} \end{cases} \quad (7\text{-}14)$$

y 称为序列 $x_1,x_2,x_3,\cdots,x_n$ 的中值。把一个点的特定长度或形状的领域称为窗口。在一维情形下，中值滤波器是一个含有奇数个像素的滑动窗口，窗口正中间那个像素的值用窗口内各像素值的中值代替。设输入序列为 $\{x_i,i\in I\}$，I 为自然数集合或子集，窗口长度为 n，则滤波器输出为

$$y_i=\mathrm{med}\{x_i\}=\mathrm{med}\{x_{i-u},\cdots,x_i,\cdots,x_{i+u}\} \quad (7\text{-}15)$$

式中，$i\in I$；$u=(n-1)/2$。

很容易将中值滤波的概念推广到二维，此时可以利用某种形式的二维窗口。设 $\{x_{ij},(i,j)\in I^2\}$ 表示数字图像各点的灰度值，滤波窗口为 A 的二维中值滤波可定义为

$$y_{ij}=\mathrm{med}_A\{x_{ij}\}=\mathrm{med}\{x_{i+r,j+s},(r,s)\in A,(i,j)\in I^2\} \quad (7\text{-}16)$$

二维中值滤波可以取方形，也可以取近似圆形或十字形。

中值滤波是非线性运算，因此对于随机性质的噪声输入，数学分析是相当复杂的。由大量实验可得，对于零均值正态分布的噪声输入，中值滤波输出与输入噪声的密度分布有关，输出噪声方差与输入噪声密度函数的平方成反比。

对随机噪声的抑制能力，中值滤波性能要比平均值滤波差些，但对于脉冲干扰来讲，特别是脉冲宽度较小，相距较远的窄脉冲，中值滤波是很有效的。

4．非线性平滑滤波器的 MATLAB 应用

下例对含有椒盐噪声的图像实现中值滤波处理（见图 7-22）。

```
clear all;
I=imread('F:\image\lena.bmp');
figure,imshow(I);
I1=imnoise(I,'salt & pepper',0.06); %加噪
K1=medfilt2(I,[3,3]); %使用 3×3 模板完成中值滤波
K2=medfilt2(I,[5,5]); %使用 5×5 模板完成中值滤波
K3=medfilt2(I,[7,7]); %使用 7×7 模板完成中值滤波
figure,imshow(K1);
```

```
figure,imshow(K2);
figure,imshow(K3);
```

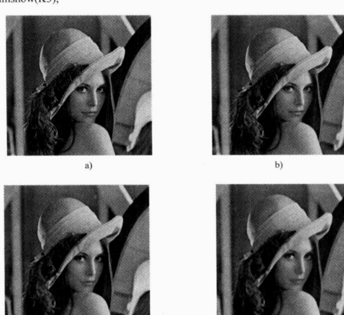


图 7-22 调用函数 medfilt2 实现中值滤波

a) 原始图像 b) 3×3 模板中值滤波 c) 5×5 模板中值滤波 d) 7×7 模板中值滤波



由以上处理结果可以看出，中值滤波器不像均值滤波那样，它在衰减噪声的同时不会使图像的边界模糊，这也是中值滤波器受欢迎的主要原因。中值滤波器去噪声的效果依赖于两个要素：领域空间范围，中值计算中所涉及的像素数。一般来说，小于中值滤波器面积一半的亮或暗的物体基本上会被滤掉，而较大的物体则几乎会原封不动地保存下来，因此中值滤波器的空间尺寸必须根据面临的问题来进行调整。较简单的模板是 $N\times N$ 的方形（这里 N 通常是奇数），计算时用到所有的（N 个）像素点。另外，读者也可以使用稀疏分布的模板来节省空间。

$$\boldsymbol{domain}=\begin{pmatrix}0&0&1&0&0\\0&0&1&0&0\\1&1&1&1&1\\0&0&1&0&0\\0&0&1&0&0\end{pmatrix}\quad \boldsymbol{domain}=\begin{pmatrix}1&0&0&0&1\\0&1&0&1&0\\0&0&1&0&0\\0&1&0&1&0\\1&0&0&0&1\end{pmatrix}\tag{7-17}$$

这里仅举两例，例中都是对于 5×5 的模板来说的，当然，实际应用中可以根据不同的情况选取不同大小的模板，来达到更好的应用效果。

需要说明的是，中值滤波只是排序统计滤波中的一种，即用当前窗口灰度排序在中间的值代替当前的值。在实现中也可以用其他规定点的值代替，在 MATLAB 中这种滤波器可以



用 ordfilt2(A,order, domain)函数来实现，分别举如下例：

B=ordfilt2(A, 1, ones (3,3))实现 3×3 的最小值滤波器，因为它取全 1 模板中排在最小位置处的那个像素。

B= ordfilt2(A, 1,[0 1 0; 1 0 1; 0 1 0])的输出是每个像素的东、西、南、北四个方向相邻像素灰度的最小值，因为它取 4-邻域的模板中排在最小位置处的那个像素。

7.2.3 锐化滤波

在图像识别中，需要有边缘鲜明的图像，即图像锐化。图像锐化的目的是为了突出图像的边缘信息，加强图像的轮廓特征，以便于人眼的观察和机器的识别。因此，从图像增强的目的看，它是与图像平滑相反的一类处理。

图像中对象的边缘像素都是变化较大的地方。而边缘模糊、线条不均是由于减少了边缘亮度差异的缘故。从数学观点来看，检查图像某区域内灰度的变化就是微分的概念，因此可以通过微分的方法进行图像锐化。根据微分方法是否线性，可将图像锐化分为线性锐化和非线性锐化两类，下面分别介绍。

1. 线性锐化滤波

线性高通滤波器是最常用的线性锐化滤波器，这种滤波器的中心系数都是正的，而周围的系数都是负的。对 3×3 的模板来说，典型的系数取值是

[0 −1 0; −1 4 −1 0 −1 0]

事实上这是拉普拉斯算子。

拉普拉斯算子是实线性导数运算，对被运算的图像它满足各向同性的要求，这对于图像增强是非常有利的。拉普拉斯算子的表达式是

$$\nabla^2 f(i,j)=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2} \tag{7-18}$$

对于离散函数 $f(i,j)$，其差分形式是

$$\nabla^2 f(i,j)=\Delta x^2 f(i,j)+\Delta y^2 f(i,j) \tag{7-19}$$

这里 $\Delta x^2 f(i,j)$ 和 $\Delta y^2 f(i,j)$ 是 $f(i,j)$ 在 x 方向和 y 方向的二阶差分，所以离散函数的拉普拉斯算子的表达式为

$$\nabla^2 f(i,j)=f(i+1,j)+f(i-1,j)+f(i,j+1)+f(i,j-1)-4f(i,j)$$

系数取值是

[0 −1 0; −1 8 −1; 0 −1 0]

1）在 MATLAB 中可通过调用 filter2( )函数和 fspecial( )函数来实现，代码如下：

```
clear all;
I1=imread('F:\image\lena.bmp');
%将图像矩阵转化为 double 型
I1=double(I1);
```

```
h1=fspecial('laplacian');
I2=filter2(h1,I1); %拉氏变换
figure,imshow(I1,[]);
I3=I1-I2;
figure,imshow(I3,[]);
```



处理前后的图像如图 7-23 所示。



a)



b)

图 7-23 拉普拉斯图像增强

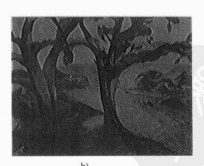
a) 原始图像 b) 拉普拉斯增强后的图像

2）下面为线性锐化滤波后面的边缘检测的例子，程序代码如下，运行结果如图 7-24 所示。

```
%下面是利用拉普拉斯算子对模糊图像进行增强
I=imread('trees.tif');
I=double(I);                %转换数据类型为 double 双精度型
figure,imshow(I,[]);
H=[0 1 0,1 -4 1,0 1 0];     %拉普拉斯算子
J=conv2(I,H,'same');        %用拉普拉斯算子对图像进行二维卷积运算
%增强的图像为原始图像减去拉普拉斯算子滤波的图像
K=I-J;
figure,imshow(K,[])
```



a)



b)

图 7-24 拉普拉斯算子对模糊图像进行增强

a) 原始图像 b) 拉普拉斯算子对图像的锐化

运行结果如图 7-24 所示。由图可见，图像模糊的部分得到了锐化，边缘部分得到了增强，边界更加明显。但图像显示清楚的地方，经滤波后发生了失真，这也是拉普拉斯算子增强的一大缺点。

2. 非线性锐化滤波

对一幅图像施加梯度模算子，可以增强灰度变化的幅度，因此我们可以采用梯度模算子作为图像的锐化算子。此方法也是最常用的非线性锐化滤波方法，而且由数学知识可以知道，梯度模算子具有方向同性和位移不变性，这正是我们所希望的。

对于离散函数 $f(i,j)$，利用差分来代替微分。

一阶差分的定义为

$$\Delta_x f(i,j) = f(i,j) - f(i-1,j) \tag{7-20}$$

$$\Delta_y f(i,j) = f(i,j) - f(i,j-1) \tag{7-21}$$

因此，梯度的定义为

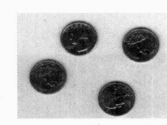
$$|G| = \left[\Delta_x f(i,j)^2 - \Delta_y f(i,j)^2\right]^{\frac{1}{2}} \tag{7-22}$$

为了运算简便，实际中采用梯度模的近似形式，如 $|\Delta_x f(i,j)| + |\Delta_y f(i,j)|$ 、$\max(|\Delta_x f(i,j)|, |\Delta_y f(i,j)|)$ 、 $\max|f(i,j) - f(m,n)|$ 等。另外，还有一些常用的算子，如 Roberts 算子和 Sobel 算子。

利用 Sobel 算子对图像滤波，程序代码如下：

```
I=imread('eight.tif');
H=fspecial('sobel');   %选择 Sobel 算子
figure,imshow(I);
J=filter2(H,I);        %卷积运算
figure,imshow(J)
```

运行结果如图 7-25 所示。


a)

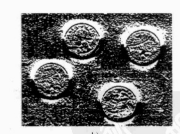

b)

图 7-25 Sobel 算子对图像锐化的结果

a) 原始图像 b) 进行 Sobel 算子后的效果

7.3 频域滤波增强

频域滤波增强方法是将图像从空间域变换到频域，在图像的频域空间对图像进行滤波处理。根据信号分析理论，傅里叶变换和卷积理论是频域滤波技术的基础，因此，在频域空间

的滤波与空间域滤波一样可以通过卷积运算实现。

假定$g(i,j)$表示函数$f(i,j)$与线性移不变算子$h(i,j)$进行卷积运算的结果，即

$$g(x,y)=f(x,y)\otimes h(x,y) \tag{7-23}$$

因此可得

$$G(u,v)=F(u,v)H(u,v) \tag{7-24}$$

式中，G、F、H 分别是函数$g(i,j)$、$f(i,j)$、$h(i,j)$的傅里叶变换；$H(u,v)$称为滤波器函数，也可以称为传递函数。在图像增强中，由于待增强的图像函数$f(i,j)$是已知的，因此，$F(u,v)$可由图像的傅里叶变换得到。

实际应用中，首先需要确定$H(u,v)$，然后就可以求得$G(u,v)$，再对$G(u,v)$进行傅里叶逆变换，即可得到增强的图像$g(i,j)$。$g(i,j)$可以突出$f(i,j)$的某一方面的特征信息。若通过$H(u,v)$增强$F(u,v)$的高频信息，如增强图像的边缘信息等，则为高通滤波；如果增强$F(u,v)$的低频信息，如对图像进行平滑操作等，则为低通滤波。频域滤波方法的系统框图如图 7-26 所示，其滤波处理过程可以分为以下三个步骤：

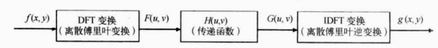


图 7-26 频域滤波系统框图

1）对原始图像$f(i,j)$进行傅里叶变换得到$F(u,v)$。

2）将$F(u,v)$与滤波器函数$H(u,v)$进行卷积运算得到$G(u,v)$。

3）对$G(u,v)$进行傅里叶逆变换，即可求出增强图像$g(i,j)$。

7.3.1 低通滤波

图像从空间域变换到频域后，其低频分量对应图像中灰度值变化比较缓慢的区域，而高频分量则表示了图像中物体的边缘和随机噪声信息。低通滤波器的功能是通过滤波器函数 H 减弱或抑制高频分量，保留低频分量。因此，低通滤波器与空域中的平滑滤波器一样可以消除图像中的随机噪声，削弱边缘效应，起到平滑图像的作用。

常用的低通滤波器包括理想低通滤波器、巴特沃斯低通滤波器、指数低通滤波器和梯形低通滤波器等多种类型。本章只讨论径向对称的零相移滤波器函数，几种常用的低通滤波器形式如下。

1．理想低通滤波器

二维的理想低通滤波器的传递函数为

$$H(u,v)=\begin{cases}1 & D(u,v)\leqslant D_0\\0 & D(u,v)>D_0\end{cases} \tag{7-25}$$

式中，D_0是一个非负整数，即理想低通滤波器的截止频率；$D(u,v)$是从点(u,v)到频域原点的距离，即

$$D(u,v)=\sqrt{u^2+v^2} \tag{7-26}$$

因此，$H(u,v)$、u、v组成了理想低通滤波器的三维图形。图 7-27a、b 分别为理想低通滤波器的三维透视图和二维剖面示意图，理想低通滤波器的作用是将小于 D_0 的频率，即以 D_0 为半径的圆内的所有频率成分可以无衰减通过，而大于 D_0 的频率则被完全截止不能通过。

理论上给出的滤波器函数（包括高通滤波）形式都是以坐标原点径向对称的，而对于一个数字图像所对应的 $N \times N$ 频域矩阵，坐标原点是该矩形的中心，因而滤波器理想特性一般如图 7-27 所示。理想低通滤波器的数学定义形式非常简洁，其平滑作用的物理意义非常明显，但在图像处理过程中会产生比较严重的模糊与振铃现象。因为，根据傅里叶变换的性质，若 $H(u,v)$ 这理想的矩形特性，那么其逆变换 $h(x,y)$ 的特性会产生无限的振铃特性，$h(x,y)$ 与 $f(x,y)$ 卷积运算后将给目标图像 $g(x,y)$ 造成模糊与振铃现象。而且 D_0 越小，这种现象越明显。

此外在截止频率 D_0 处垂直截止的理想低通滤波器只能通过计算机模拟实现，无法采用电子器件实现。

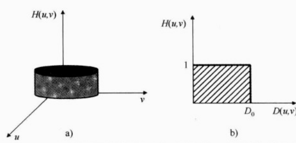


图 7-27　理想低通滤波器的三维透视图和二维剖面示意图

a) 理想低通滤波器的三维透视图　b) 理想低通滤波器的二维剖面示意图

2. 巴特沃斯低通滤波器

巴特沃斯（Butterworth）低通滤波器的传递函数为

$$H(u,v)=\frac{1}{1+\left[\dfrac{D(u,v)}{D_0}\right]^{2n}} \tag{7-27}$$

式中，D_0 为截止频率；n 为滤波器的阶次。和理想低通滤波器一样，巴特沃斯低通滤波器的特性曲线同样为三维图形，其剖面示意图如图 7-28 所示。一般情况下，当 $H(u,v)$ 下降到最大值的 1/2 时，$D(u,v)$ 为截止频率 D_0。在实际应用中，有时也取 $H(u,v)$ 下降至最大值的 $\sqrt{2}/2$ 时的 $D(u,v)$ 作为截止频率 D_0。这时，其传递函数为

$$H(u,v)=\frac{1}{1+\left(\sqrt{2}-1\right)\left[\dfrac{D(u,v)}{D_0}\right]^{2n}} \tag{7-28}$$

巴特沃斯低通滤波器又称为最大平坦滤波器，其通带与阻带之间的过渡比较平坦。因此，巴特沃斯低通滤波器的特点是：在通过频率与截止频率之间没有明显的不连续性，不会出现“振铃”效应，其效果好于理想低通滤波器。

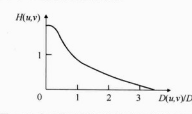


图 7-28 巴特沃斯低通滤波器的剖面示意图

3. 指数低通滤波器

指数低通滤波器的传递函数为

$$H(u,v)=\mathrm{e}^{-\left[\frac{D(u,v)}{D_0}\right]^n} \tag{7-29}$$

一般情况下，取 $H(u,v)$ 下降至最大值的 1/2 时的 $D(u,v)$ 为截止频率 D_0。其剖面示意图如图 7-29 所示。与巴特沃斯低通滤波器一样，指数低通滤波器从通过频率到截止频率之间具有一段平滑的过渡带，也没有明显的不连续性。

4. 梯形低通滤波器

梯形低通滤波器的传递函数为

$$H(u,v)=\begin{cases}1 & D(u,v)<D_0\\ \dfrac{D(u,v)-D_1}{D_0-D_1} & D_0\leqslant D(u,v)\leqslant D_1\\ 0 & D(u,v)>D_1\end{cases} \tag{7-30}$$

梯形低通滤波器的剖面图示意如图 7-30 所示，从图中可以看出，在 D_0 的尾部包含有一部分高频分量（$D_1>D_0$），因而，结果图像的清晰度较理想低通滤波器有所改善，“振铃”效应也有所减弱，应用时可调整 D_1 的值，即能达到平滑图像的目的，又可以使图像保持足够的清晰度。

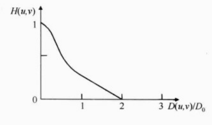


图 7-29 指数低通滤波器的剖面示意图

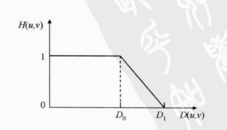


图 7-30 梯形低通滤波器的剖面示意图

根据 DFT 变换的性质和频谱的分布特点，二维 DFT 变换的频谱主要集中在低频处，因此，在设计和应用低通滤波器时，一定要注意二维 DFT 的频谱特点，即图像能量集中在频谱图的频谱中心位置。

5. 频域实现图像平滑滤波

各种频域低通滤波器的 MATLAB 实现，程序代码如下：

```
[I,map]=imread('F:\image\lena.bmp');
noisy=imnoise(I,'gaussian',0.01);
[M,N]=size(I);
F=fft2(noisy);
fftshift(F);
Dcut=100;
D0=150;
D1=250;
for u=1:M
    for v=1:N
        D(u,v)=sqrt(u^2+v^2);
        BUTTERH(u,v)=1/(1+(sqrt(2)-1)*(D(u,v)/Dcut)^2);
        EXPOTH(u,v)=exp(log(1/sqrt(2))*(D(u,v)/Dcut)^2);
        if D(u,v)<D0
            THPFH(u,v)=1;
        elseif D(u,v)<=D1
            THPEH(u,v)=(D(u,v)-D1)/(D0-D1);
        else
            THPFH(u,v)=0;
        end
    end
end
BUTTERG=BUTTERH.*F;
BUTTERfiltered=ifft2(BUTTERG);
EXPOTG=EXPOTH.*F;
EXPOTfiltered=ifft2(EXPOTG);
THPFG=THPFH.*F;
THPFfiltered=ifft2(THPFG);
figure,imshow(noisy);
figure,imshow(BUTTERfiltered,map)
figure,imshow(EXPOTfiltered,map)
figure,imshow(THPFfiltered,map);
```

程序运行结果如图 7-31 所示。





a)





b)





c)

d)

图 7-31　频域低通滤波举例

a) 高斯噪声后的图像　b) 巴特沃斯低通滤波后的图像　c) 指数低通滤波后的图像　d) 梯形低通滤波后的图像

7.3.2 高通滤波

图像中物体的边缘及其他灰度变化较快的区域与图像的高频信息有关，因此利用高通滤波器可以对图像的边缘信息进行增强，起到锐化图像的作用。高通滤波器包括理想高通滤波器、巴特沃斯高通滤波器、指数高通滤波器和梯形高通滤波器等类型。本章只讨论径向对称的零相移滤波器函数，几种常用的高通滤波器形式如下。

1．理想高通滤波器

二维理想高通滤波器的传递函数为

$$H(u,v)=\begin{cases}0 & D(u,v)\leqslant D_0\\ 1 & D(u,v)>D_0\end{cases} \tag{7-31}$$

式中，D_0是一个非负整数，即理想高通滤波器的截止频率；$D(u,v)$是从点(u,v)到频域原点的距离，即

$$D(u,v)=\sqrt{u^2+v^2} \tag{7-32}$$

图 7-32 中给出了理想高通滤波器滤波特性的剖面示意图，其作用与理想低通滤波器相反，它将小于D_0的频率（半径为D_0的圆内）的所有频率完全截止，而大于D_0的频率（圆外的频率）则可以全部无衰减通过。理想高通滤波器也不能通过电子器件实现。

2．巴特沃斯高通滤波器

巴特沃斯高通滤波器的传递函数为

$$H(u,v)=\frac{1}{1+\left[\dfrac{D_0}{D(u,v)}\right]^{2n}} \tag{7-33}$$

式中，D_0 为滤波器的截止频率；n 为滤波器的阶次。巴特沃斯高通滤波器滤波性能的剖面示意图如图 7-33 所示。截止频率 D_0 的取值方法与巴特沃斯低通滤波器相似，该滤波器在通过频率与截止频率之间也没有明显的不连续性，图像增强后，“振铃”效应不明显。

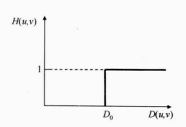


图 7-32　理想高通滤波器的剖面示意图

图 7-33　巴特沃斯高通滤波器的剖面示意图

与巴特沃斯低通滤波器类似，一般情况下，取 $H(u,v)$ 下降至最大值的 1/2 时的 $D(u,v)$ 为截止频率 D_0。在实际应用中，有时也取 $H(u,v)$ 下降至最大值的 $\sqrt{2}/2$ 时的 $D(u,v)$ 作为截止频率 D_0。这时，其传递函数形式为

$$H(u,v)=\frac{1}{1+\left(\sqrt{2}-1\right)\left[\dfrac{D}{D_0(u,v)}\right]^{2n}} \tag{7-34}$$

3．指数高通滤波器

指数高通滤波器的传递函数为

$$H(u,v)=\mathrm{e}^{-\left[\frac{D_0}{D_0(u,v)}\right]^{n}} \tag{7-35}$$

其截止频率 D_0 的取值与指数低通滤波器相似，其特性曲线剖面示意图如图 7-34 所示。

4．梯形高通滤波器

梯形高能滤波器的传递函数为

$$H(u,v)=\begin{cases}1 & D(u,v)<D_0\\ \dfrac{D(u,v)-D_1}{D_0-D_1} & D_1\leqslant D(u,v)\leqslant D_0\\ 0 & D(u,v)<D_1\end{cases} \tag{7-36}$$

如图 7-35 所示为梯形高通滤波器滤波的剖面示意图。

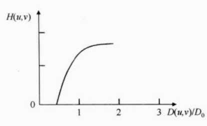


图 7-34 指数高通滤波器的剖面示意图

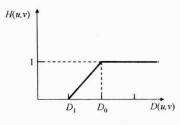


图 7-35 梯形高通滤波器的剖面示意图

7.3.3 带通和带阻滤波器

在图像处理中，有时需要增强的信息或抑制的信息既不是图像中的高频成分也不是低频成分，而是在一个有限的频带范围内。这时，无论是低通滤波器还是高通滤波器都不能完全满足使用需求，而需要采用带通或带阻滤波器。

1．带通滤波器

所谓带通滤波器是指允许一定频率范围内的信号通过而阻止其他频率范围内的信号通过的滤波器。理想带通滤波器的传递函数为

$$H(u,v)=\begin{cases}0 & D(u,v)<D_0-\dfrac{w}{2}\\ 1 & D_1-\dfrac{w}{2}\leqslant D(u,v)\leqslant D_0+\dfrac{w}{2}\\ 0 & D(u,v)<D_1+\dfrac{w}{2}\end{cases} \tag{7-37}$$

式中，w 为通带宽度；D_0 为通带中心频率；$D(u,v)$ 表示从点 (u,v) 到频带中心 (u_0,v_0) 的距离，即

$$D(u,v)=\sqrt{(u-u_0)^2+(v-v_0)^2} \tag{7-38}$$

理想带通滤波器的剖面示意图如图 7-36 所示。

2．带阻滤波器

带阻滤波器是指可以对一定频率范围内的信号进行完全衰减，而容许其他频率范围内的信号通过的滤波器。理想带阻滤波器的传递函数为

$$H(u,v)=\begin{cases}1 & D(u,v)<w_1\\ 0 & w_1<D(u,v)<w_2\\ 1 & D(u,v)\leqslant w_2\end{cases} \tag{7-39}$$

式中，w_1,w_2 为带阻宽度；$D(u,v)$ 表示从点 (u,v) 到带阻中心 (u_0,v_0) 的距离，即

$$D(u,v)=\sqrt{(u-u_0)^2+(v-v_0)^2} \tag{7-40}$$

理想带阻滤波器的剖面示意图如图 7-37 所示。

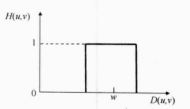


图 7-36 理想带通滤波器的剖面示意图

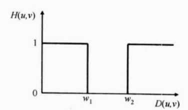


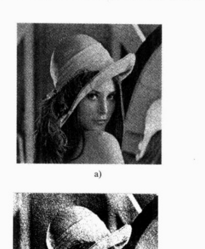
图 7-37 理想带阻滤波器的剖面示意图

7.3.4 频域滤波的 MATLAB 实现

如下用频域高通滤波法对图像进行增强。程序代码如下：

```
[I,map]=imread('F:\image\lena.bmp');
noisy=imnoise(I,'gaussian',0.01);
[M,N]=size(I);
F=fft2(noisy);
fftshift(F);
Dcut=100;
D0=250;
D1=150;
for u=1:M
    for v=1:N
        D(u,v)=sqrt(u^2+v^2);
        BUTTERH(u,v)=1/(1+(sqrt(2)-1)*(Dcut/D(u,v))^2);
        EXPOTH(u,v)=exp(log(1/sqrt(2))*(Dcut/D(u,v))^2);
        if D(u,v)<D1
            THPFH(u,v)=0;
        elseif D(u,v)<=D0
            THPEH(u,v)=(D(u,v)-D1)/(D0-D1);
        else
            THPFH(u,v)=1;
        end
    end
end
BUTTERG=BUTTERH.*F;
BUTTERfiltered=ifft2(BUTTERG);
EXPOTG=EXPOTH.*F;
EXPOTfiltered=ifft2(EXPOTG);
THPFG=THPFH.*F;
THPFfiltered=ifft2(THPFG);
figure,imshow(noisy);
figure,imshow(BUTTERfiltered)
figure,imshow(EXPOTfiltered)
figure,imshow(THPFfiltered);
```

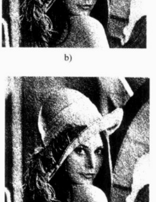
程序代码执行后的结果如图 7-38 所示。



a)



b)



c)

d)

图 7-38　频域高通滤波举例

a) 加入高斯噪声后的图像　b) 巴特沃斯高通滤波后的图像　c) 指数高通滤波后的图像　d) 梯形高通滤波后的图像

7.4　同态增晰

同态增晰是一种在频域中将图像动态范围进行压缩并将图像对比度进行增强的方法。它基于图像的成像模型。一幅图像 $f(x,y)$ 可以用它的照明分量 $i(x,y)$ 及反射分量 $r(x,y)$ 来表示，即

$$f(x,y)=i(x,y)r(x,y) \tag{7-41}$$

根据这个模型可用下列方法对两个分量分别进行滤波，如图 7-39 所示。

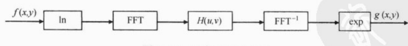


图 7-39　同态增晰流程图

1）先对式（7-41）取对数，$\ln f(x,y)=\ln i(x,y)+\ln r(x,y)$。

2）对式（7-41）取傅里叶变换，$F(u,v)=I(u,v)+R(u,v)$。

3）用一个频域函数 $H(u,v)$ 处理 $F(u,v)$，$H(u,v)\ F(u,v)=H(u,v)\ I(u,v)+H(u,v)\ R(u,v)$。

4）反变换到空间域，$h_f(x,y)=h_i(x,y)+h_r(x,y)$。

5）将 4）中的式子两边取指数，$g(x,y)=\exp|h_f(x,y)|=\exp|h_i(x,y)|\exp|h_r(x,y)|$，令

$$i_0(x,y)=\exp|h_i(x,y)|$$
$$r_0(x,y)=\exp|h_r(x,y)|$$

则

$$g(x,y)=i_0(x,y)r_0(x,y)$$

式中，$i_0(x,y)$ 是处理后的照射分量；$r_0(x,y)$ 是处理后的反射分量。

一幅图像的照射分量一般是在空间缓慢变化的，而反射分量在不同物体的交界处是急剧变化的，这个特征使人们有可能把一幅图像取对数后的傅里叶变换的低频分量和照射分量联系起来，而把反射分量与高频分量联系起来。以上特性表明我们可以设计一个对傅里叶变换的高频分量和低频分量影响不同的滤波函数 $H(u,v)$，处理结果会使像素灰度的动态范围或图像对比度得到增强。如图 7-40 所示，低频段被压缩，高频段得到增强，结果是同时压缩了图像的动态范围并增加了图像各部分之间的对比度。如图 7-41 所示为同态增晰滤波增强效果。

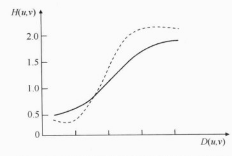
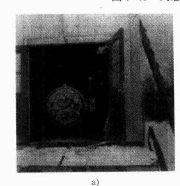
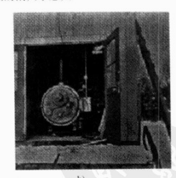


图 7-40 同态增晰滤波函数的剖面示意图



a)



b)

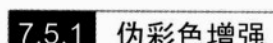
图 7-41 同态增晰滤波增强效果

a) 光照不均的原因　b) 同态增晰后的效果图

7.5 彩色图像增强

前面介绍的图像增强技术都是对灰度图像进行处理的，而且生成的结果也是灰度图像。本节的彩色增强技术处理的对象虽然也是灰度图像，但生成的结果却是彩色图像。众所周知，人

的视觉系统对色彩非常敏感，人眼一般能区分的灰度等级只有二十多个，但是能区分有不同亮度、色度和饱和度的几千种颜色。根据人眼的这个特点，可将彩色用于图像增强中，以提高图像的可鉴别性。因此如果能将一幅灰度图像变成彩色图像，就可以达到图像增强的视觉效果。常用的彩色增强方法有真彩色增强技术、假彩色增强技术和伪彩色增强技术三种。前两种方法着眼于对多幅灰度图像的合成处理，一般是将三幅图像分别作为红、绿、蓝三个通道进行合成。伪彩色增强技术与前两者不同，它是对一幅灰度图像的处理，通过一定的方法，将一幅灰度图像变成一幅彩色图像。下面分别对伪彩色、真彩色和假彩色增强技术进行详细论述。



7.5.1 伪彩色增强

伪彩色(Pseudo Coloring)增强是把一幅黑白图像的每个不同灰度级，按照线性或非线性的映射函数，变换成不同的彩色，与彩色空间中的一点相匹配，得到一幅彩色图像的技术。它使原图像细节更易辨认、目标更容易识别。伪彩色增强的方法主要有以下三种。

1. 密度分割法

密度分割法也称为强度分割法，是伪彩色增强中最简单而又最常用的一种方法，它是对图像的灰度值动态范围进行分割，使分割后的每个灰度值区间甚至每个灰度值本身对应某一种颜色，如图 7-42a、b 所示。具体而言，假定把一幅图像看成一个二维的强度函数，用于一个平行于图像坐标平面的平面（称为密度切割平面）去切割图像的强度函数，这样强度函数在分割处被分为上、下两部分，即两个灰度值区间。如果再对每一个区间赋予某种颜色，就可以将原来的灰度图像变换成只有两种颜色的图像。更进一步，如果用多个密度切割平面对图像的强度函数进行分割，那么就可以将图像的灰度值动态范围切割成多个区间，每一个区间赋予某一种颜色，则原来的一幅灰度图像就可以变成一幅彩色图像，如图 7-42 所示。其中，图 7-42a 以立体方式给出了密度分割原理的立体图，图 7-42b 是平面图。特别地，如果将每一个灰度值都划分成一个区间，如将 8b 灰度图像划分成 256 个区间，就是索引色图像，从这个意义上讲，可以认为索引色图像是由灰度图像经密度分割而成的。

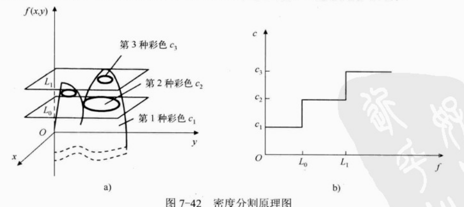


图 7-42　密度分割原理图

a) 密度分割原理的立体图　b) 密度分割原理的平面图

如果用 N 个密度切割平面去切割图像的强度函数，则可以得到 $N+1$ 个灰度值区间，每一个区间对应一种颜色 C_i。对于每个像元 (x,y)，如果 $I_{i-1} \leqslant f(x,y) \leqslant I_i$，则 $g(x,y)=C_i$，$i=1,2,\cdots,N$，$g(x,y)$ 和 $f(x,y)$ 分别表示变换后的彩色图像和原始灰度图

像。这样便可以把一幅灰度图像变成一幅伪彩色图像。此法比较直观简单，缺点是变换出的彩色数目有限。

应当指出，每个灰度值区间赋予何种颜色，是由具体应用所决定的，并无规律可言。但总的来讲，相邻灰度值区间的颜色差别不宜太小，也不宜太大，太小将无法反映细节上的差异；太大则会导致图像出现不连续性。在实际应用中，密度切割平面之间可以是等间隔的，也可以是不等间隔的，而且密度切割平面的划分也应根据具体的应用范围和研究对象而定。

2．空间域灰度级—彩色变换

密度分割法实质上是通过一个分段线性函数实现从灰度到彩色的变换，每个像元只经过一个变换对应到某一种颜色。与密度分割法不同的是，空间域灰度级—彩色变换是一种更为常用、比密度分割法更有效的伪彩色增强法。其变换过程如图 7-43 所示，它是根据色度学的原理，将原始图像 $f(x,y)$ 中每个像元的灰度值分别经过红、绿、蓝三种独立变换 $T_R(\cdot)$、$T_G(\cdot)$ 和 $T_B(\cdot)$，变成红、绿、蓝三基色分量 $R(x,y)$、$G(x,y)$、$B(x,y)$ 分量图像，然后用它们分别去控制彩色显示器的红、绿、蓝电子枪，便可以在彩色显示器的屏幕上合成一幅彩色图像。三个变换是独立的，彩色的含量由变换函数 $T_R(\cdot)$、$T_G(\cdot)$ 和 $T_B(\cdot)$ 的形状而定。但在实际应用中，这三个变换函数一般取同一类的函数，如可以取带绝对值的正弦函数，也可以取线性变换函数。典型的变换函数如图 7-44 所示，灰度值范围为 $[0,L]$，每个变换取不同的分段线性函数。可以看出，最小的灰度值（0）对应蓝色，中间的灰度值（$L/2$）对应绿色，最高的灰度值（L）对应红色，其余的灰度值则分别对应不同的颜色。其中图 7-44a～c 分别为红、绿、蓝三种变换函数，而图 7-44d 是把三种变换画在同一张图上，以便看清楚它们之间的关系。由图 7-44d 可见，只有在灰度为零时，呈蓝色；灰度为 $L/2$ 时，呈绿色；灰度为 L 时，呈红色；灰度为其他值时，将由三基色混合成不同的色调。

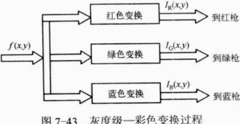


图 7-43　灰度级—彩色变换过程

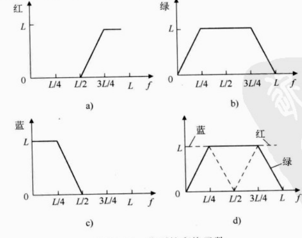


图 7-44　典型的变换函数

a) 红色　b) 绿色　c) 蓝色　d) 红、绿、蓝三色关系

3. 频域伪彩色增强

频域伪彩色增强首先把黑白图像从空间域经傅里叶变换变到频域，然后在频域内用三个不同传递特性的滤波器（如高通、带通/带阻、低通）将图像分离成三个独立的分量，对每个范围内的频率分量分别进行傅里叶逆变换，得到三幅代表不同频率分量的单色图像，接着对这三幅图像作进一步的处理（如直方图均衡化），最后将它们作为三基色分量分别加到彩色显示器的红、绿、蓝显示通道，从而实现频域的伪彩色增强，如图 7-45 所示。

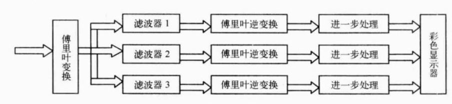


图 7-45　频域伪彩色增强原理图

7.5.2 假彩色增强

假彩色（False Coloring）增强是将一幅图像或多光谱图像映射到 RGB 空间中心位置上的过程。假彩色增强是经常出现的一个操作过程。例如，调节彩色电视机的色调、饱和度的过程实际就是假彩色增强。又如红光成像设备拍摄了 N 幅不同波段上的图像：$f_1(x,y)$，$f_2(x,y),\cdots,f_N(x,y)$，将它们经过假彩色增强可以再现出可见光谱图像，其处理函数如下：

$$R(x,y)=F_R[f_1(x,y),f_2(x,y),\cdots,f_N(x,y)] \tag{7-42}$$

$$G(x,y)=F_G[f_1(x,y),f_2(x,y),\cdots,f_N(x,y)] \tag{7-43}$$

$$B(x,y)=F_B[f_1(x,y),f_2(x,y),\cdots,f_N(x,y)] \tag{7-44}$$

式中，F_R,F_G,F_B 为映射函数；$R(x,y),G(x,y),B(x,y)$ 为显示空间三基色分量。

伪彩色或假彩色增强都不改变图像像素的几何位置，而仅仅改变其颜色。因此，可以与人眼的色觉特性相结合设计它们的映射函数 F_R,F_G,F_B，提高人眼对图像的分辨能力。该技术已被广泛应用于遥感、医学图像处理中。

7.5.3 真彩色增强

自然物体的彩色叫真彩色，把能真实反映自然物体本来颜色的图像叫真彩色图像。真彩色图像可由彩色摄像机摄制，并由彩色监视器近似复原。然而，在没有彩色摄像机的情况下，也可以通过真彩色增强技术实现真彩色处理。

任何一幅真彩色图像可由红、绿、蓝三基色混合而成。因此，在图像处理过程中，首先用加有红色滤色片的摄像机（黑白摄像机）摄取彩色图像，图像信号经数字化送入一块图像存储板存起来；再用带有绿色滤色片的摄像机摄取图像，图像信号经数字化送入第二块图像存储板；最后用带有蓝色滤色片的摄像机摄取图像，图像数据存储在第三块图像存储板内。三幅图像数据准备好后，就可以在系统的输出设备——彩色监视器上合成一幅真彩色图像。其原理如图 7-46 所示。

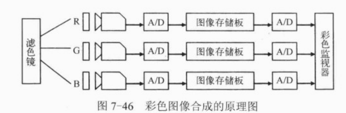


图 7-46 彩色图像合成的原理图

习题

7-1 假定有 64×64 大小的图像，灰度级为 16 级，概率分布见表 7-1，试进行直方图均衡化并画出处理前后的直方图。

表 7-1

r	n_k	$p_r(r_k)$	r	n_k	$p_r(r_k)$
r_0=0	800	0.195	r_8=8/15	150	0.037
r_1=1/15	650	0.160	r_9=9/15	130	0.031
r_2=2/15	600	0.147	r_{10}=10/15	110	0.027
r_3=3/15	430	0.106	r_{11}=11/15	96	0.013
r_4=4/15	300	0.073	r_{12}=12/15	80	0.019
r_5=5/15	230	0.056	r_{13}=13/15	70	0.017
r_6=6/15	200	0.049	r_{14}=14/15	50	0.012
r_7=7/15	170	0.041	r_{15}=1	30	0.007

7-2 比较理想高通滤波器与理想低通滤波器的异同点。

7-3 如果一幅图像已经用直方图均衡化方法进行了处理，那么对处理后的图像再次应用直方图均衡化，处理结果会不会更好？

7-4 比较理想低通滤波器、巴特沃斯低通滤波器、指数低通滤波器和梯形低通滤波器。

7-5 为什么一般情况下对离散图像直方图均衡化并不能产生完全平坦的直方图？

7-6 已知一幅图像的灰度级为 8 级，即（0，1）之间划分为 8 个灰度等级。图像的左边一半为深灰色，其灰度级为 1/7，而右边一半是黑色，其灰度级为 0，如图 7-47 所示。试对此图像进行直方图均衡化处理，并描述一下处理后图像是一幅什么样的图像。


图 7-47

7-7 试证明拉普拉斯算子$\left(\frac{\partial^2}{\partial x^2}\right)+\left(\frac{\partial^2}{\partial y^2}\right)$具有旋转不变性。

7-8 什么是伪彩色增强处理？其主要目的是什么？

第 8 章　图像分割与边缘检测



图像分割是一种重要的图像分析技术。在对图像的研究和应用中，人们往往仅对图像中的某些部分感兴趣，这些部分常常被称为目标或前景（其他部分称为背景），它们一般对应图像中特定的、具有独特性质的区域。这里的独特性质可以是像素的灰度值、物体轮廓曲线、颜色和纹理等。为了识别和分析图像中的目标，需要将它们从图像中分离提取出来，在此基础上才有可能进一步对目标进行测量和对图像进行利用。图像分割就是指图像分成各具特性的区域并提取出感兴趣目标的技术和过程。

一般的图像处理过程如图 8-1 所示。从图中可以看出，图像分割是从图像预处理到图像识别和图像分析理解的关键步骤，在图像处理中占据重要的位置。一方面它是目标表达的基础，对特征测量有重要的影响；另一方面，图像分割以及基于图像分割的目标表达、特征提取和参数测量等将原始图像转化为更为抽象、更为紧凑的形式，使得更高层的图像识别和图像分析理解成为可能。

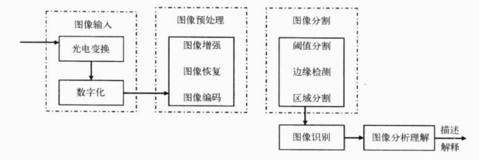


图 8-1　一般的图像处理过程

图像分割的方法已有上千种，每年还有许多新方法出现，典型而传统的图像分割方法可以分为基于阈值的方法、基于边缘的方法和基于区域的分割方法等，本章将对这些典型的图像分割方法加以介绍。

8.1　灰度阈值法

8.1.1　图像分割基本原理

图像分割是根据图像的组成结构和应用需求将图像划分成若干个互不相交的子区域的过程。这些子区域是某种意义下具有共同属性的像素的连通集合。如不同目标物体所占的图像区域、前景所占的图像区域等。连通是指集合中任意两点之间都存在着完全属于该集合的连

通路径。

对于数字图像而言，如图 8-2 所示，连通包括 4 连通和 8 连通两种情况。4 连通是指从区域内一点出发，可通过 4 个方向，即上、下、左、右移动的组合，在不超过区域的前提下，到达区域内的任意像素点；8 连通是指从区域内一点出发，可通过左、右、上、下、左上、右上、左下、右下这 8 个方向的移动组合，到达区域内的任意像素点。

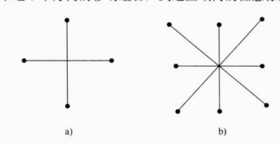


图 8-2　4 连通和 8 连通示意图

a) 4 连通　b) 8 连通

根据上述基本概念，可以给出图像分割的一般性定义，即图像分割是指将一幅离散数字图像信号 $f(m,n)$ 进行分割，将 f 分割为若干相连的、非空子区域 $f_1, f_2, f_3, \cdots, f_n$，且满足如下均一性准则。

1） $f_1 \cup f_2 \cup f_3 \cup \cdots \cup f_n = f$。

2） $\forall_i$，当 $i = 1,2,3,\cdots,n$ 时，f_i 是相连的。

3） $\forall\ f_i$ 均一性准则都是满足的。

4）对于任意两个相连的 f_i 和 f_i，$E(f_i \cup f_j) = \phi$。

目前，已经提出的图像分割方法很多，综合各种方法的实质，图像分割有三种不同的途径，分别介绍如下。

1）以区域为对象进行分割，以相似性原则作为分割的依据，即根据图像的灰度、色彩、变换关系或组织结构等方面的特征相似来划分图像的子区域，并将各像素划归到相应物体或区域的像素聚类方法，即区域法。

2）以物体的边界为对象进行分割，通过直接确定区域间的边界来实现分割的边界方法。

3）先检测边缘像素，再将边缘像素连接起来构成边界形成分割。

这些方法是互补的，在有些场合适宜应用某一种分割方法，而另一些场合则适宜采用另一种分割方法，有时还要将它们有机地结合起来，以求得更好的分割效果。

值得指出的是，图像分割没有唯一的、标准的、通用的分割方法，也没有一成不变、适应一切情况的最优分割准则，而必须根据具体图像和不同应用目的来采用合适的分割方法。在图像分割技术中，最常用的是利用图像阈值化处理进行的图像分割。

8.1.2 灰度阈值法分割

常用的图像分割方法是把图像灰度分成不同的等级，然后用设置灰度门限值（阈值）的方法确定有意义的区域或分割物体的边界。常用的阈值化处理就是图像的二值化处理，即选

择一个阈值，将图像转换为黑白二值图像，用于图像分割及边缘提取等处理之中。

图像阈值化处理的变换函数形式如下：

$$g(x,y)=\begin{cases}0 & f(x,y)\leqslant T\\ 255 & f(x,y)>T\end{cases} \tag{8-1}$$

显然，图像阈值化处理是一种阶梯函数，属于图像灰度级的非线性运算，该变换函数曲线如图 8-3 所示。它的功能是由用户指定一个阈值，如果图像中某个像素的灰度值大于该阈值，则将该像素的灰度值置为 255，否则将其灰度值置为 0。

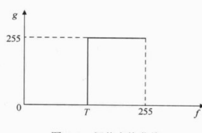
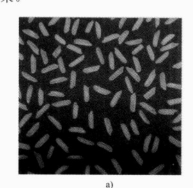
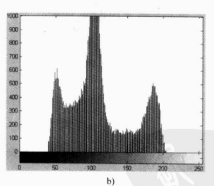


图 8-3　阈值变换曲线

采用图像的阈值化分割过程中，阈值的选取对处理结果的影响很大。如图 8-4 所示，图 8-4a 为原始图像，图像中的目标为米粒，图 8-4b 为原始图像的灰度直方图。分析该直方图可知，该直方图具有双峰特性，图像中的目标分布在较暗的灰度级上形成一个波峰，图像背景分布在较亮的灰度级上形成另一个波峰。因此，从理论上讲，以直方图双峰之间的谷底处灰度值作为阈值进行图像的阈值化处理，便可将目标和背景分割开来。



a)



b)

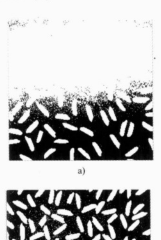
图 8-4　阈值分割法原理

a) 原始图像 b) 直方图

图 8-5a 选取阈值 $T=91$；图 8-5b 选取阈值 $T=140$；图 8-5c 选取阈值 $T=120$；图 8-5d 选取阈值 $T=56$，由于选取了不同的阈值 T，因此，图像的分割结果具有明显的差别。程序代码如下，结果如图 8-5 所示。

```
clear
```

```
I=imread('rice.png');
I2=im2bw(I,91/255);
I3=im2bw(I,140/255);
I4=im2bw(I,120/255);
I5=im2bw(I,56/255);
figure,imshow(I2)
figure,imshow(I3)
figure,imshow(I4)
figure,imshow(I5)
```



a)



b)



c)



d)

图 8-5　不同阈值对阈值化结果的影响

a) T=91　b) T=140　c) T=120　d) T=56

由图 8-5 可知，图像分割过程中，若阈值选取过大，则会提取图像的多余部分；若阈值过小，则又会丢失所需的部分。因此，阈值的选取对图像的分割具有非常重要的作用。

当图像灰度直方图的双峰特性不明显，即图像中的目标部分和背景之间亮度差较小时，直接用直方图就不太容易确定一个合适的阈值。此时，可用最小误差阈值法、最大误差阈值法、最佳阈值法、判别法分析法或其他合理的方法来确定阈值。

1．最小误差阈值

假如一幅图像，设对象物的灰度分布具有平均值为μ，标准差为δ的正态分布概率密度函数$p(z)$；背景的灰度分布具有平均值为ν，标准差为τ的正态分布概率函数$q(z)$，如图 8-6 所示。

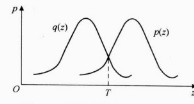


图 8-6　目标和背景概率密度分布

设对象物占整体图像的比例为t，此时整体图像的灰度概率密度为

$$tp(z)+(1-t)q(z) \tag{8-2}$$

现在用阈值 T 分开：当$z<T$时为背景，反之则是对象物。此时，把背景误认为对象物的概率为

$$Q(T)=\int_{\theta}^{+\infty}q(z)\mathrm{d}z \tag{8-3}$$

把对象物误认为背景的概率为

$$1-P(T)=\int_{-\infty}^{\theta}p(z)\mathrm{d}z \tag{8-4}$$

那么错误区分的概率为

$$t[1-P(T)]+(1-t)Q(T) \tag{8-5}$$

求式（8-5）为最小值时的θ，便是阈值，也就是对式（8-5）求微分并使其为零。

$$\frac{\mathrm{d}}{\mathrm{d}T}\{t[1-P(T)]+(1-t)Q(T)\}=0$$

所以

$$(1-t)q(T)-tp(T)=0 \tag{8-6}$$

根据假设，当 t、$p(z)$、$q(z)$ 已知时，可求解阈值T。利用这种方法求取阈值 T 时，必须用两个已知正态分布的曲线合成来近似直方图的分布，还要给定两个正态分布合成的比例 t，所以实现起来比较复杂，必须用数值计算才能得到。

2. 最大方差阈值

最大方差阈值也叫大津阈值，是 1980 年由日本的大津展之提出的，它是在差别与最小二乘法原理的基础上推导出来的，可得到较好的结果。

把直方图在某一阈值处分割成两组，当被分成的两组间方差为最大时，决定阈值。现在，设一幅图像的灰度值为 1～m 级，灰度值 i 的像素数为 n_i，此时得到：

像素总数为

$$N=\sum_{i=1}^{m}n_i \tag{8-7}$$

各灰度值的概率为

$$p_i=\frac{n_i}{N} \tag{8-8}$$

然后用 T 将其分成两组 $C_0=\{1\sim T\}$ 和 $C_1=\{T+1\sim m\}$，各组产生的概率如下：

C_0 产生的概率为

$$w_0=\sum_{i=1}^{T}p_i=w(T) \tag{8-9}$$

C_1 产生的概率为 $$w_1 = \sum_{i=T+1}^{m} p_i = 1 - w_0 \quad (8-10)$$

C_0 的平均值为 $$\mu_0 = \sum_{i=1}^{T} \frac{ip_i}{w_0} = \frac{\mu(T)}{w(T)} \quad (8-11)$$

C_1 的平均值为 $$\mu_1 = \sum_{i=T+1}^{m} \frac{ip_i}{w_1} = \frac{\mu - \mu(T)}{1 - w(T)} \quad (8-12)$$

式中，$\mu = \sum_{i=1}^{m} ip_i$ 是整体图像的灰度平均值；$\mu(T) = \sum_{i=1}^{T} ip_i$ 是阈值为 T 时的灰度平均值，所以全部采样的灰度平均值为

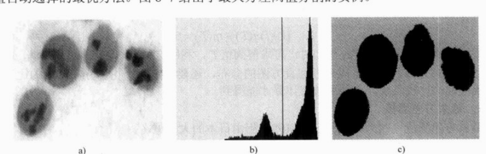
$$\mu = w_0\mu_0 + w_1\mu_1 \quad (8-13)$$

两组间的方差可用下式求出：

$$\begin{aligned}\delta^2(T) &= w_0(\mu_0 - \mu)^2 + w_1(\mu_1 - \mu)^2 = w_0 w_1(\mu_1 - \mu_0) \\ &= \frac{[\mu w(T) - \mu(T)]^2}{w(T)[1 - w(T)]}\end{aligned} \quad (8-14)$$

从 1～m 改变 T，求式（8-14）为最大值时的 T，即求 $\max \delta^2(T)$ 时的 T^* 值，此时，T^* 便是阈值。$\delta^2(T)$ 叫做阈值选择函数。

不管图像的直方图有无明显的双峰，此方法都能得到较满意的结果，因此，这种方法是阈值自动选择的最优方法。图 8-7 给出了最大方差阈值分割的实例。



a)　　b)　　c)

图 8-7　最大方差阈值分割实例

a) 原始图像　b) 最大方差阈值选择　c) 阈值分割结果

3. 最佳阈值法

设图像由物体和背景两部分组成，物体像素的灰度级具有正态概率密度函数 $p(z)$，其均值为 μ，方差为 δ^2。背景像素的灰度级也具有正态概率密度函数 $q(z)$，其均值为 υ，方差为 τ^2。物体占总图像面积的比例为 λ，背景所占面积为 $1-\lambda$，因此，该图像总的灰度概率密度函数为

$$\lambda p(z) + (1-\lambda)q(z) \quad (8-15)$$

令图像阈值为 t，并且将小于 t 的全部点称为目标物体点，而将不小于 t 的全部点称为背景点。那么，设将背景点错归为目标物体点的概率为 $Q_1(t)$，将目标物体点错归为背景点的概率为 $Q_2(t)$，因而有

$$Q_1(t)=\int_{-\infty}^{t} q(z)\mathrm{d}z \tag{8-16}$$

$$Q_2(t)=\int_{t}^{\infty} p(z)\mathrm{d}z=1-P(t) \tag{8-17}$$

总的错分概率为

$$\lambda Q_2(t)+(1-\lambda)Q_1(t)=\lambda[1-P(t)]+(1-\lambda)Q_1(t) \tag{8-18}$$

显然，使上式最小的阈值t为最佳阈值。因此，对式（8-18）微分可得

$$(1-\lambda)q(t)=\lambda p(t) \tag{8-19}$$

由于$p(t)$和$q(t)$都服从正态分布，因而有

$$p(t)=\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{(t-\mu)^2}{2\sigma^2}} \tag{8-20}$$

$$q(t)=\frac{1}{\sqrt{2\pi}\tau}\mathrm{e}^{-\frac{(t-\upsilon)^2}{2\tau^2}} \tag{8-21}$$

将式（8-20）、（8-21）代入式（8-19）并取自然对数可得

$$\ln\sigma+\ln(1-\lambda)-\frac{(t-\upsilon)^2}{2\tau^2}=\ln\tau+\ln\lambda-\frac{(t-\mu)^2}{2\sigma^2}$$

$$\tau^2(t-\mu)^2-\sigma^2(t-\upsilon)^2=2\sigma^2\tau^2\ln\frac{\tau\lambda}{\sigma(1-\lambda)}$$

根据上式，若$\lambda=0.5,\tau=\sigma$，则可以得出：

$$t=\frac{\mu+\upsilon}{2} \tag{8-22}$$

4. 差别分析法

差别分析法确定最佳阈值的原则是进行阈值处理后，被分离的像素类之间的类间方差最大。差别分析法需要计算直方图的0阶矩和1阶矩，它是图像阈值化处理中常用的自动确定阈值的方法。

设图像总像素数为N，灰度值为i的像素数以N_i表示，则灰度级为K的灰度分布的0阶矩及1阶矩分别定义如下。

0阶矩为

$$\omega(k)=\sum_{i=0}^{k}\frac{N_i}{N} \tag{8-23}$$

1阶矩为

$$\omega(k)=\sum_{i=1}^{k}\frac{iN_i}{N} \tag{8-24}$$

当$k=L-1$时，$\omega(L-1)=1$；$\mu(L-1)=\mu_A$，其中，μ_A为图像的平均灰度值。

设图像的$M-1$个阈值为：$0\leqslant k_1<k_2<k_3<\cdots<k_{M-1}\leqslant L-1$，将图像分割成$M$个灰度值的类$C_j$，且

$$C_j\in[k_{j-1}+1,\cdots,k_j]\quad k_0=0,k_M=L,j=1,2,3,\cdots,M$$

则各类C_j发生的概率为

$$\omega_j=\omega(k_j)-\omega(k_{j-1})$$

而相应的概率平均值为

$$\mu_j = \frac{\mu(k_j) - \mu(k_{j-1})}{\omega(k_j) - \omega(k_{j-1})} \tag{8-25}$$

其中，$\omega(0) = 0, \mu(0) = 0$。

因此，可得各类的类间方差为

$$\sigma^2(k_1, k_2, k_3, \cdots, k_{M-1}) \sum \omega_j (\mu_j - \mu_T)^2 \tag{8-26}$$

式中，μ_T为相应的概率值。

将使式（8-26）的σ^2值为最大的阈值组$(k_1, k_2, k_3, \cdots, k_{M-1})$，作为 M 值化的最佳阈值组。若分割成两类，即取 M 为 2，则可求出二值化的阈值。

对于复杂图像，在许多情况下对整幅图像用单一阈值不能给出理想的分割结果。例如，若为光亮背景上的暗物体图像，由于照射光的不均匀，虽然物体与背景始终有反差，但在图像的某一部分物体和背景都比另一部分亮。因此，在图像的一部分能把物体和背景精确地分开的阈值，对另一部分而言，可能会把太多的背景也作为物体进行分割。克服这一缺点有如下一些方法：如果已知图像上的位置函数用来描述不均匀照射，则可以先设法应用灰度级校正方法进行校正，然后采用单一阈值进行分割；另一种方法是将图像分成子区域，对每一子区域设置局部阈值。但是，如果某一子区域仅仅含有物体或仅有背景，那么该子区域就找不到阈值。这时，可以根据邻近子区域的局部阈值通过内插值法求得该子区域的一个阈值。

在确定阈值时，如果阈值过高，则偶然出现的物体像素点就会被认为是背景点；如果阈值过低，则会发生相反的情况。克服这种情况的方法是使用两个阈值。例如，$t_1 < t_2$，将灰度值超过t_2的像素分类为核心物体点，而灰度值超过t_1的像素，仅当它们紧靠核心物体点时才认为是物体点。t_2的选择要使每个物体有一些像素灰度级高于t_2，而背景不含有这样的像素。同时，应选择t_1使每个物体像素点高于t_1灰度级。如果只使用t_2，则物体总是分割得不完整；如果只使用t_1，则会有许多背景像素被错分为物体像素。但是，如果同时使用t_1和t_2两个值，就能把背景与物体很好地分割开来。当然，如果物体与背景的对比是鲜明的，就不必使用这种方法。

此外，如果存在一个阈值 t_2，使得每个物体的像素灰度级高于 t_2，而背景不包含这种像素，则可对图像设置阈值 t_2，然后检查高于阈值像素的区域，目的是寻找一个局部阈值，以便在每个类似区域中把物体和背景分开。如果这些物体相当小，并且不太靠近，那么这种方法比较适用。所使用的区域应足够大，以保证它们既包含物体像素，也包含背景像素，这样就可以使区域的直方图是双峰的。

有时需要寻找一幅图像的局部最大点，即提取比附近像素具有较高的某种局部性质值的像素。一般来讲，也要求这些点具有高于一个低阈值 t_1 的值，一旦超过 t_1，不管它的绝对值大小如何，一切相对的最大值都被采纳。因此，可把寻找局部最大值看为局部设置阈值的极端情况。在对图像进行匹配运算或检测界线时可采用这种方法。

8.2 边缘检测

数字图像的边缘检测是图像分割、目标区域识别、区域形状提取等图像分析领域十分重

要的基础，也是图像识别中提取图像特征的一个重要属性。在进行图像理解和分析时，第一步往往就是边缘检测，目前它已成为机器视觉研究领域最活跃的课题之一，在工程应用中占有十分重要的地位。

物体边缘是以图像的局部特征不连续的形式出现的，即是指图像局部亮度变化最显著的部分，如灰度值的突变、颜色的突变、纹理结构的突变等，同时物体的边缘也是不同区域的分界处。图像边缘具有方向和幅度两个特性，通常沿边缘的走向灰度变化平缓，垂直于边缘走向的像素灰度变换剧烈。根据灰度变化的特点，可分为阶跃型、房顶型和凸缘型，如图 8-8 所示。

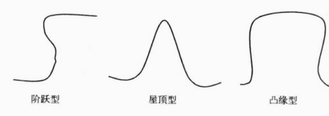


图 8-8 图像边缘的灰度变化

利用边缘检测来分割图像，其基本思想就是先检测图像中的边缘点，再按照某种策略将边沿点连接成轮廓，从而构成分割区域。由于边缘是所要提取目标和背景的分界线，提取出边缘才能将目标和背景分开，因此边缘检测技术对于数字图像十分重要。

图像中某物体边界上的像素点，其领域将是一个灰度级变化带。衡量这种变化最有效的两个特征值就是灰度的变化率和变化方向，它们分别以梯度向量的幅值和方向来表示。对于连续图像 $f(x,y)$，其方向导数在边缘（法线）方向上有局部最大值。因此，边缘检测就是求 $f(x,y)$ 梯度的局部最大值和方向。

已知 $f(x,y)$ 在 θ 方向沿 r 的梯度定义如下：

$$\frac{\partial f}{\partial r}=\frac{\partial f}{\partial x}\frac{\partial x}{\partial r}+\frac{\partial f}{\partial y}\frac{\partial y}{\partial r}=f_x\cos\theta+f_y\sin\theta \tag{8-27}$$

$\frac{\partial f}{\partial r}$ 达到最大值的条件是 $\frac{\partial\left(\partial f/\partial r\right)}{\partial\theta}=0$，即

$$-f_x\sin\theta_g+f_y\cos\theta_g=0$$

$$得\theta_g=\arctan f_y/f_x,或\ \theta_g+\pi \tag{8-28}$$

梯度最大值 $g=\left(\frac{\partial f}{\partial r}\right)_{\max}=\sqrt{f_x^2+f_y^2}$，一般称为梯度模。梯度模算子具有位移不变性和各向同性的性质，适用于边缘检测，而灰度变化的方向，即边界的方向则可由 $\theta_g=\arctan\theta f_y/f_x$ 得到。

在实际应用中，为了简便，一般将算子以微分算子的形式表示，然后采用快速卷积函数来实现，这种实现方法可以得到快速而有效的处理。

8.2.1 微分算子

常用的微分算子有 Roberts 算子、Prewitt 算子、Sobel 算子以及 Isotropic Sobel 算子，这里主要介绍前三个算子。

1. Roberts 算子

对于离散图像来说，边缘检测算子就是用图像的垂直和水平差分来逼近梯度算子：

$$\nabla f=(f(x,y)-f(x-1,y)-f(x,y-1)) \tag{8-29}$$

因此，当需要检测图像边缘时，最简单的方法就是对每个像素计算 ∇f，然后求绝对值，最后进行阈值操作就可以实现。Roberts 算子就是基于这种思想，该算子见式（8-30）。

$$R(i,j)=\sqrt{(f(i,j)-f(i+1,j+1))^2+(f(i,j+1)-f(i+1,j))^2} \tag{8-30}$$

它可以由以下两个 2×2 的模板共同实现：

$$\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \qquad \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}$$

2. Prewitt 算子和 Sobel 算子

在比较复杂的图像中，仅用 2×2 的 Roberts 算子得不到较好的边缘检测，而相对较复杂的 3×3 的 Prewitt 算子和 Sobel 算子检测效果较好。

和 Roberts 算子类似，Prewitt 算子也可以通过以下两个模板实现：

$$\begin{pmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{pmatrix} \qquad \begin{pmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{pmatrix}$$

以上两矩阵分别代表图像的水平梯度和垂直梯度。如果用 Prewitt 算子检测图像 M 的边缘，一般先用水平算子和垂直算子对图像进行卷积，得到两个矩阵 $M1$、$M2$，在不考虑边界因素的时候，它们与原图像有相同的大小，分别表示图像 M 中相同位置对 P_V 和 P_H 的偏导数。然后求 $M1$ 和 $M2$ 对应位置的两个数的平方和，得到一个新的矩阵 G。G 是 M 中像素灰度梯度的近似值，然后经过阈值操作得到边缘，即

$$G=((M\otimes P_V)^2+(M\otimes P_H)^2)>Thresh^2$$

Sobel 算子与 Prewitt 算子的区别仅在于选用的模板不同：

$$\begin{pmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{pmatrix} \qquad \begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix}$$

不同的算子选取不同的模板，为什么要这么选取呢？这是由以下原理决定的。

假定图像 M 的灰度满足以下关系式：

$$M_{x,y}=\alpha x+\beta y+\gamma$$

即梯度为 (α,β)，则每一像素的 8 领域像素值为

$$\begin{pmatrix} -\alpha-\beta+\gamma & -\alpha+\gamma & -\alpha+\beta+\gamma \\ -\beta+\gamma & \gamma & \beta+\gamma \\ \alpha-\beta+\gamma & \alpha+\gamma & \alpha+\beta+\gamma \end{pmatrix}$$

则定义水平算子和垂直算子为

$$\begin{pmatrix} -a & 0 & a \\ -b & 0 & b \\ -a & 0 & a \end{pmatrix} \qquad \begin{pmatrix} -a & -b & -a \\ 0 & 0 & 0 \\ a & b & a \end{pmatrix}$$

将这两个模板同原始图像像素进行卷积，可得到的方向导数为

$$g_x = 2\beta(2a+b)$$
$$g_y = 2\alpha(2a+b)$$

所以得到像素的梯度大小为

$$g = \left(\frac{\partial f}{\partial r}\right)_{\max} = \sqrt{g_x^2 + g_y^2} = 2(2a+b)\sqrt{\alpha^2+\beta^2} \tag{8-31}$$

显然，如果要使得梯度为常量，则应该使得 $2(2a+b)=1$。

如果 $a=b=1/6$，则得到 1/6 乘 Prewitt 算子；取 $a=1/8$，$b=1/4$，则得到 1/8 乘 Sobel 算子。

为了方便，下面对上述常用算子的模板进行总结，见表 8-1。

表 8-1 常用边缘检测算子模板

算子名称	H_1	H_2	特点
Roberts	$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	边缘定位准 对噪声有抑制作用
Prewitt	$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	平均、微分 对噪声有抑制作用
Sobel	$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$	加树平均 边宽≥2 像素
Isotropic Sobel	$\begin{pmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ -1 & \sqrt{2} & 1 \end{pmatrix}$	树值反比于邻点与中心点的距离 检测沿不同边缘方向时梯度幅度一致

MATLAB 7.0 图像处理工具箱中提供了专门的边缘检测 edge()函数，其调用格式如下：

- BW=edge(I, 'method')
- BW=edge(I, 'method', thresh)
- BW=edge(I, 'method', thresh, direction)
- [BW, thresh]=edge(I, 'method',⋯)

其中，I 是输入图像，method 是选用的方法（算子），可以选择的 method 有 Sobel、Prewitt、Roberts、log、Candy、zerocross 等。

可选的参数有 thresh（门限）、sigma（方差）和 direction（方向）。

下面，将利用 edge()函数，分别采用 3 种不同的边缘检测算子对图 8-9a 所示的原始图像进行边缘提取，程序代码如下：

```
I=imread('tire.tif');
figure,imshow(I);
BW1=edge(I,'sobel',0.1);
figure,imshow(BW1)
BW2=edge(I,'roberts',0.1)
figure,imshow(BW2)
BW3=edge(I,'prewitt',0.1)
figure,imshow(BW3)
```

执行以上程序，效果如图 8-9b～d 所示。

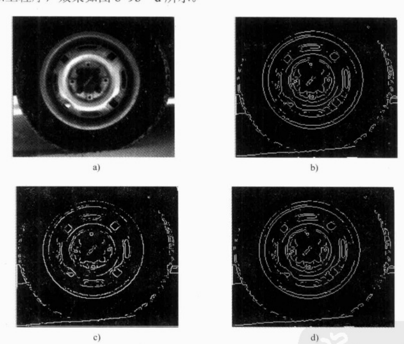


图 8-9　不同的边缘检测算子进行提取的效果

a) 原始图像　b) Sobel 算子检测效果　c) Roberts 算子检测效果　d) Prewitt 算子检测效果

比较 3 个算子的检测效果，可以发现 Sobel 算子和 Prewitt 算子的效果比较好。

8.2.2 拉普拉斯高斯算子（LOG）

前面都是利用边缘处的梯度最大（正的或者负的）这一性质来进行边缘检测的，即利用了灰度图像的拐点位置是边缘的性质。除了这一点，边缘还有另外一个性质，即在拐点位置处的二阶导数为 0，如图 8-10 所示。

图 8-10　边缘、局部极值、零交叉点

图中由上到下分别是图像的拐点（图像灰度值）、拐点处的梯度（灰度一阶导数）和灰度二阶导数。对准图像网格点，可以发现二阶导数为零交叉点处对应的是图像的拐点。

所以，也可以通过寻找二阶导数的零交叉点来寻找边缘，而 Laplacian 算子是最常用的二阶导数算子。

二元函数 $f(x,y)$ 的 Laplacian 变换定义为

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (8\text{-}32)$$

实际上就是二阶偏导数的和。将上式以差分方式表示，得到式（8-33）：

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)] \qquad (8\text{-}33)$$

以模板形式表示，就得到了常用的算子：

$$\nabla^2 f = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$\nabla^2 f$ 算子能突出反映图像中的角线和孤立点，如对图 8-11a 所示的原始数据图像进行 Laplacian 算子运算，可以得到如图 8-11b 所示的结果，在边缘和孤立点的幅值都比较大。

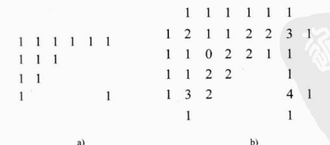


图 8-11　原始数据图像及运算结果

a) 原始数据图像　b) 用 Laplacian 算子运算以后的结果

但是，需要注意的是，由上述算子可以知道，一阶导数对噪声敏感，因而不稳定，由

此，二阶导数对噪声就会更加敏感，因而更不稳定。所以，在进行 Laplacian 变换之前需要作平滑。同时，又因为卷积是可变换、可结合的，所以先作高斯卷积，再用 Laplacian 算子作卷积，等价于对原始图像用高斯函数的 Laplacian 变换后的滤波器作卷积。这样就得到一个新的滤波器——LOG（Laplacian Of Gaussian）滤波器。

$$f(x,y)=\nabla^2(G(x,y)*M(x,y)) \tag{8-34}$$

式中，$M(x,y)$ 是图像。

$$G(x,y)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{8-35}$$

$$\mathrm{LOG}(x,y)=\nabla^2(G(x,y))=\frac{\partial^2 G}{\partial x^2}+\frac{\partial^2 G}{\partial y^2}$$

$$=\frac{-1}{2\pi\sigma^4}\left(2-\frac{x^2+y^2}{\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{8-36}$$

利用以下程序代码段，可以得到 LOG 算子的图像，如图 8-12 所示。

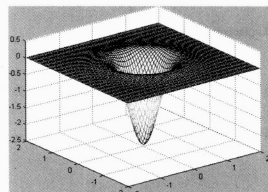


图 8-12　LOG 算子图像

```
clear;
x=-2:0.06:2;
y=-2:0.06:2;
sigma=0.6;
y=y';
for i=1:(4/0.06+1)
    xx(i,:)=x;
    yy(:,i)=y;
end
r=1/(2*pi*sigma^4)*((xx.^2+yy.^2)/(sigma^2)-2).*exp(-(xx.^2+yy.^2)/(sigma^2));
colormap(jet(16));
mesh(xx,yy,r)
```

由图 8-12 可以看出，它是一个带通滤波器，研究表示，LOG 算子比较符合人的视觉特性。在实际应用中，使用 LOG 模板作卷积，然后寻找那些零交叉像素：如果一个像素值小于

$-\theta_0$，而周围邻接的 8 个像素都大于θ_0，则这个像素就是零交叉点。利用以下程序代码，对图 8-9a 所示的原始图像进行 LOG 算子边缘提取，得到如图 8-13 所示的效果。

```
BW=edge(I,'log',0.01)
figure,imshow(BW)
```



图 8-13　LOG 算子边缘提取的结果



8.2.3 Canny 算子

还有一个很重要的边缘检测算子，即 Canny 算子，它是最优的阶梯型边缘（Step Edge）检测算子。从以下 3 个标准意义来说，Canny 边缘检测算子对受到白噪声影响的阶跃型边缘是最优的。

1）检测标准，不丢失重要的边缘，不应有虚假的边缘。

2）定位标准，实际边缘与检测到的边缘位置之间的偏差最小。

3）单响应标准，将多个响应降低为单个边缘响应。

Canny 算子的实现步骤如下：

1）首先用 2D 高斯滤波模板与原始图像进行卷积，以消除噪声。

2）利用导数算子（如 Prewitt 算子、Sobel 算子）找到图像灰度沿着两个方向的导数 G_x,G_y，并求出梯度的大小：$|G|=\sqrt{G_x^2+G_y^2}$。

3）利用 2）的结果计算出梯度的方向：$\theta=\arctan\left(\dfrac{G_y}{G_x}\right)$。

4）求出了边缘的方向，就可以把边缘的梯度方向大致分为 4 种（0°、45°、90°和 135°），并可以找到这个像素梯度方向的邻接像素。

5）遍历图像。若某个像素的灰度值与其梯度方向上前后两个像素的灰度值相比不是最大的，那么将这个像素值置为 0，即不是边缘。

6）使用累计直方图计算两个阈值。凡是大于高阈值的一定是边缘；凡是小于低阈值的一定不是边缘。如果检测结果在两个阈值之间，则根据这个像素的邻接像素中有没有超过高阈值的边缘像素，如果有，则它就是边缘，否则不是。

以下程序段是利用 MATLAB 中的 edge()函数，并采用 Canny 算子，对如图 8-14a 所示的原始图像进行边缘检测，检测效果如图 8-14b 所示。



a)



b)

图 8-14　Canny 算子运算

a) 原始图像　b) Canny 算子边缘检测效果

```
I=imread('tire.tif');
figure,imshow(I);
BW=edge(I,'canny',0.1)
figure,imshow(BW)
```

比较图 8-14b 与前面几个算子进行边缘检测的结果，可以看出 Canny 算子检测结果不单能清晰地提取 tire 的边缘，而且边缘连续比较好，这就是 Canny 算子的优良之处。

当然，还有许多其他边缘检测算法，如与 Canny 算子类似，还有利用边缘方向性的检测算子。其算法描述如下：

1）设置 4 个 3×3 模板如图 8-15 所示。显而易见，4 个模板分别针对 0°、45°、90° 和 135° 4 个方向的梯度设置，并以 (x, y) 点为中心将 3×3 的区域分成两个部分，按照这 4 个模板分别对图像中的每一像素点进行卷积求和操作。

2）对图像中每一像素点求出的 4 个卷积和求绝对值，并将每个结果分别与一个阈值比较，如果其中任意一个结果大于或等于阈值 T，则该模板的中心点所对应的像素点的灰度值为 225，否则为 0。

-1	0	1
-1	0	1
-1	0	1

0°

0	1	1
-1	0	1
-1	-1	0

45°

1	1	1
0	0	0
-1	-1	-1

90°

0	-1	-1
1	0	-1
1	1	0

135°

图 8-15　边缘检测算子模板

8.3　区域分割

阈值分割法由于没有或很少考虑空间关系，使多阈值选择受到限制，基于区域的分割方法可以弥补这点不足。该方法利用的是图像的空间性质，认为分割出来的属于同一区域的像素应具有相似的性质，其概念是相当直观的。传统的区域分割法有区域生长法和分裂合并法，下面对这两种方法加以介绍。

8.3.1　区域生长

1. 区域生长的原理和步骤

区域生长的基本思想是将具有相似性质的像素集合起来构成区域。具体是先对每个需要分割的区域找一个种子像素作为生长的起点，然后将种子像素周围领域中与种子像素有相同或相似性质的像素（根据某种事先确定的生长或相似准则来判定）合并到种子像素所在的区域中。将这些新像素当做新的种子像素继续进行上面的过程，直到再没有满足条件的像素可被包括进来，这样，一个区域就长成了。

图 8-16 给出了已知种子点进行区域生长的一个示例。图 8-16a 给出待分割的图像，设已知有两个种子像素（标为深浅不同的灰色方块），现要进行区域生长。假设这里采用的判断准则是：如果所考虑的像素与种子像素灰度值差的绝对值小于某个阈值T，则将该像素包括进种子像素所在的区域。图 8-16b 给出 T=3 时的区域生长结果，整幅图像被较好地分成两个区域；图 8-16c 给出T=1 时的区域生长结果，有些像素无法判定；图 8-16d 给出 T=6 时的区域生长结果，整幅图像都被分在一个区域中了，由此可见阈值的选择是很重要的。

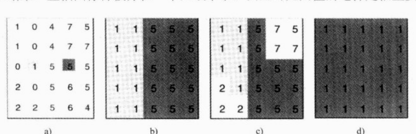


图 8-16　区域生长示例

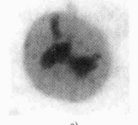
从上面的示例可知，在实际应用区域生长法时需要解决 3 个问题：

1）选择或确定一组正确代表所需区域的种子像素。

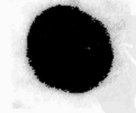
2）确定在生长过程中能将相邻像素包括进来的准则。

3）制定生长过程停止的条件或规则。

种子像素的选取常可借助具体问题的特点进行。例如，军用红外图像中检测目标时，由于一般情况下目标辐射较大，所以可以选用图中最亮的像素作为种子像素。如果对具体问题没有先验知识，则可借助生长所用准则对像素进行相应计算。如果计算结果呈现聚类的情况，则接近聚类中心的像素可取为种子像素。以图 8-17 为例，通过对它做直方图可知，具有灰度值为 1 和 5 的像素最多且处在聚类的中心，所以可各选一个具有聚类中心灰度值的像素作为种子。在选择种子像素时可以由某种规则自动选取，也可以用交互的方式完成。



a)

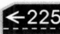


b)

图 8-17　区域生长的实例

a) 原始图像　b) 区域生长结果

生长准则的选取不仅依赖于问题本身，也和所用图像数据的种类有关。例如，当图像是彩色的时候，仅用单色的准则效果就会受到影响。另外还要考虑像素间的连通性和邻近性，否



则，有时会出现无意义的结果。我们将在后面介绍几种典型的生长准则和对应的生长过程。

一般生长过程在进行到再也没有满足生长准则的像素时停止，但常用的基于灰度、纹理和彩色的准则大都基于图像的局部性质，并没有充分考虑生长的“历史”。为增加区域生长的性能，常需考虑一些与尺寸、形状等图像和目标的全局性质有关的准则。在这种情况下，常需对分割结果建立一定的模型或辅以一定的先验知识。

2. 生长准则

区域生长的一个关键是选择合适的生长准则，大部分区域生长准则使用图像的局部性质。生长准则可根据不同原则制定，而使用不同的生长准则会影响区域生长的过程，下面介绍 3 种基本的生长准则和方法。

（1）基于区域灰度差

基于区域灰度差的方法主要有以下步骤。

1）对像素进行扫描，找出尚没有归属的像素。

2）以该像素为中心检查它的领域像素，即将领域中的像素逐个与它比较，如果灰度差小于预先确定的阈值，将它们合并。

3）以新合并的像素为中心，返回到步骤 2），检查新像素的领域，直到区域不能进一步扩张。

4）返回到步骤 1），继续扫描，直到所有像素都归属，则结束整个生长过程。

采用上述方法得到的结果对区域生长起点的选择有较大的依赖性。为克服这个问题可以对方法进行以下改进：将灰度差的阈值设为零，这样具有相同灰度值的像素便合并到一起，然后比较所有相邻区域之间的平均灰度差，合并灰度差小于某一阈值的区域。这种改进仍然存在一个问题，即当图像中存在缓慢变化的区域时，有可能会将不同区域逐步合并而产生错误的分割结果。一个比较好的做法是：在进行生长时，不用新像素的灰度值与领域像素的灰度值比较，而是用新像素所在区域的平均灰度值与各领域像素的灰度值进行比较，将小于某一阈值的像素合并进来。

（2）基于区域内灰度分布统计性质

这里考虑以灰度分布相似性作为生长准则来决定区域的合并，具体步骤为

1）把像素分成互不重叠的小区域。

2）比较邻接区域的累积灰度直方图，根据灰度分布的相似性进行区域合并。

3）设定终止准则，通过反复进行步骤 2）中的操作将各个区域依次合并，直到满足终止准则。

为了检测灰度分布情况的相似性，采用下面的方法。这里设 $h_1(X)$ 和 $h_2(X)$ 为相邻的两个区域的灰度直方图，X 为灰度值变量，从这个直方图求出累积灰度直方图 $H_1(X)$ 和 $H_2(X)$，根据以下两个准则：

① Kolomogorov-Smirnov 检测

$$\max_X |H_1(X)-H_2(X)| \tag{8-37}$$

② Smoothed-Difference 检测

$$\sum_X |H_1(X)-H_2(X)| \tag{8-38}$$

如果检测结果小于给定的阈值，就把两个区域合并。这里灰度直方图 $h(X)$ 的累积灰度

直方图 $H(X)$ 被定义为

$$H(X)=\int_0^X h(x)\mathrm{d}x$$

在离散情况下

$$H(X)=\sum_{i=0}^{X}h(i)\int_0^X h(x)\mathrm{d}x \tag{8-39}$$

对上述两种方法有两点值得说明：

① 小区域的尺寸对结果影响较大。尺寸太小，检测可靠性降低；尺寸太大，则得到的区域形状不理想，小的目标可能漏掉。

② 式（8-38）比式（8-37）在检测直方图相似性方面较好，因为它考虑了所有灰度值。

（3）基于区域形状

在决定对区域的合并时，也可以利用对目标形状的检测结果，常用的方法有两种：

1）把图像分割成灰度固定的区域，设两相邻区域的周长为 p_1 和 p_2，把两区域共同边界线两侧灰度差小于给定值的那部分设为 L，如果（T_2 为预定阈值）

$$\frac{L}{\min\{p_1,p_2\}}>T_2 \tag{8-40}$$

则合并两区域。

2）把图像分割成灰度固定的区域，设两邻接区域的共同边界长度为 B，把两区域共同边界线两侧灰度差小于给定值的那部分长度设为 L，如果（T_2 为预定阈值）

$$\frac{L}{B}>T_2 \tag{8-41}$$

则合并两区域。

上述两种方法的区别是：第一种方法是合并两相邻区域的共同边界中对比度较低部分占整个区域边界份额较大的区域，第二种方法则是合并两相邻区域的共同边界中对比度较低部分比较多的区域。

下述为区域生长的实例。

图 8-17a 是细胞图像，在细胞体上手工选择一种子点，采用区域灰度差的生长准则，图 8-17b 是区域生长的结果。

8.3.2 分裂合并

8.3.1 节介绍的区域生长法是先从单个种子像素开始通过不断接纳新像素，最后得到整个区域。分裂合并法是从整幅图像开始通过不断分裂得到各个区域。在实际中，常先把图像分成任意大小且不重叠的区域，然后再合并或分裂这些区域，以满足分割的要求。

在这类方法中，常需要根据图像的统计特性设定图像区域属性的一致性测度，其中最常用的测度多基于灰度统计特征，如同质区域中的方差（Variance Within Homogeneous Region，VWHR）。算法根据 VWHR 的数值合并或分裂各个区域。为得到正确的分割结果，需要根据先验知识或对图像中噪声的估计来选择 VWHR，它选择的精度对算法性能影响很大。

假设以 VWHR 的一致性测度，令 $V(R)$ 代表趋于区域 R 内的 VWHR 值，阈值设为 T，下面介绍一种利用图像四叉树（Quad Tree，QT）表达方法的简单分裂合并算法。如图 8-18

所示，设 R_0 代表整个四方形图像区域，从最高层开始，如果 $V(R_0)>T$，就将其四等分，得到四个子区域 R_i。如果 $V(R_i)>T$，则将该区域四等分。如此类推，直到 R_i 为单个像素。

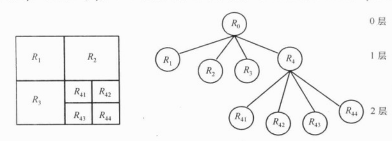


图 8-18　简单的区域分裂过程

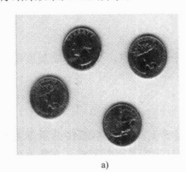
如果仅仅使用分裂，最后有可能出现相邻的两个区域属于同一个目标，但并没有合并成一个整体。为解决这个问题，每次分裂后允许其以后继续分裂或合并。合并过程只合并相邻的区域，且合并后组成的新区域要满足一致性测度，即相邻的 R_i 和 R_j，如果 $V(R_i \cup R_j) \leqslant T$，则将二者合并。

下面用 qtdecomp()函数实现四叉树分解。

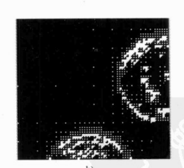
程序代码如下：

```
I=imread('eight.tif')
J=I(1:128,1:128);
S=qtdecomp(J,0.2);
figure,imshow(I);
figure,imshow(full(S))
```

程序执行结果如图 8-19 所示。



a)



b)

图 8-19　四叉树分解

a) 原始图像　b) 四叉树分解的结果

综上所述，分裂合并算法的步骤可以简单描述如下：

1）对于任意一个 R_i，如果 $V(R_i)>T$，则将其分裂成互不重叠的四等分。

2）对相邻区域 R_i 和 R_j，如果 $V(R_i \cup R_j) \leqslant T$，则将二者合并。

3）如果进一步的分裂或合并都不可能了，则终止算法。

8.3.3 水域分割

1. 水域分割的基本原理

水域分割，又称为 Watershed 变换，是一种借鉴了形态学理论的分割方法，其本质上是利用图像的区域特性来分割图像的，它将边缘检测与区域生长的优点结合起来，能够得到单像素宽、连通、封闭、且位置准确的轮廓。水域分割的基本思想是基于局部极小值和积水盆（Catchment Basin）的概念。积水盆是地形中局部极小点的影响区（Influence Zones），水平面从这些局部极小值处上涨，在水平面浸没地形的过程中，每一个积水盆被筑起的“坝”所包围，这些坝用来防止不同积水盆里的水混合到一起。在地形完全浸没到水中之后，这些筑起的坝就构成了分水岭。这个过程如图 8-20 所示。

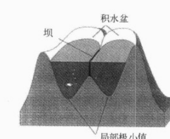


图 8-20　地形浸没过程说明

现在将水域的概念应用到图像分割中，假设待分割的图像由目标和背景组成，这样，图像的背景和目标的内部区域将对应梯度图中灰度较低的位置，而目标边缘则对应了梯度图中的亮带，称梯度图像中具有均匀低灰度值的区域为极小值区域（一般分布在目标内部及背景处）。水面从这些极小区域开始上涨，当不同流域中的水面不断升高到将要汇合在一起时（目标边界处），便筑起一道堤坝，最后得到由这些坝组成的分水线，图像也就完成了分割。图 8-21 是一个水域分割实例。

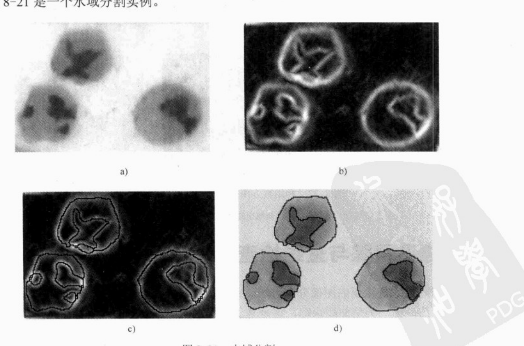


图 8-21　水域分割

a) 原始图像　b) 经典的 Canny 梯度　c) 梯度图像的流域分界线　d) 分割结果

2．基于标记的 Watershed 变换

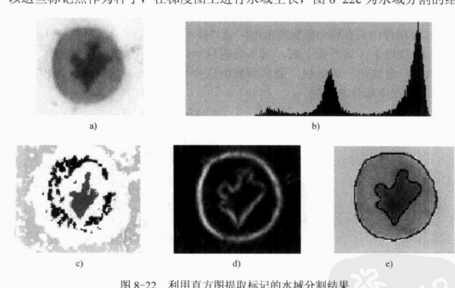
由于待分割的图像中存在噪声和一些微小的灰度值起伏波动，在梯度图像中可能存在许多假的局部极小值，如果直接对梯度图像进行生长会造成过分割的现象。即使在 Watershed 变换前对梯度图像进行滤波，存在的极小点也往往会多于原始图像中目标的数目，因此必须加以改进。实际中应用 Watershed 变换的有效途径是首先确定图像中目标的标记或种子，然后再进行生长，并且在生长的过程中仅对具有不同标记的标记点建筑防止溢流汇合的坝，产生分水岭，这就是基于标记的 Watershed 变换。基于标记的 Watershed 变换大体可分为 3 个步骤：

1）对原始图像进行梯度变换，得到梯度图。

2）用合适的标记函数把图像中相关的目标及背景标记出来，得到标记图。

3）将标记图中的相应标记作为种子点，对梯度图像进行 Watershed 变换，产生分水岭。

由于目标标记的正确与否直接影响分割结果，所以利用 Watershed 变换进行图像分割的关键是标记提取。到目前为止，标记提取还没有一个统一的方法，一般依赖于图像的先验知识，如图像极值、平坦区域或纹理等。图 8-22a～d 是一种利用直方图峰值特性提取标记的方法，算法将直方图中的 3 个峰所对应的像素作为标记，分成对应核仁、细胞核及背景 3 类目标，以这些标记点作为种子，在梯度图上进行水域生长，图 8-22e 为水域分割的结果。



a) b) c) d) e)

图 8-22 利用直方图提取标记的水域分割结果

a) 原始图像 b) 直方图 c) 提取出的标记 d) Canny 梯度 e) 水域分割的结果

8.4 边界跟踪与直线检查

边界跟踪技术是重要的图像分割方法。它分为两类：一类是区域跟踪，这是基于区域的图像分割方法；另一类是曲线跟踪，这是基于边界的图像分割方法。

由于直线通常对应重要的边缘信息，直线提取是计算机视觉中一项非常重要的技术。如在车辆自动驾驶技术分析中，道路的提取需要有效地提取直的道路边缘；在航空照片分析中，直线对应于重要的人造目标的边缘。因此，把直线提取单独抽出来进行研究很有意义。

8.4.1 基本原理

曲线跟踪的基本思路是：从当前的一个 “现在点” （边缘点）出发，用跟踪准则检查“现在点”的邻点，满足跟踪准则的像素点被接受为新的“现在点”并做上标记。在跟踪过程中可能出现以下几种情况：“现在点”是曲线的分支点或几条曲线的交点，取满足跟踪准则的各邻点中的一个点作为新的“现在点”，继续进行跟踪，而将其余满足跟踪准则的各邻点存储起来，供以后继续跟踪用；当跟踪过程中的“现在点”的邻点都不满足跟踪准则时，则该分支曲线跟踪结束。当全部分支点处的全部待跟踪的点均已跟踪完毕后，该次跟踪过程结束。

跟踪准则除了可能使用灰度值、梯度模值之外，还可能使用平滑性要求。另外，起始点的选择和搜索准则的确定对曲线跟踪的结果影响很大。

区域跟踪也称为区域生长，它的基本思路是：在图像中寻找满足某种检测准则（如灰度门限）的点，对任意一个这样的点，检查它的全部邻点，把满足跟踪准则的任何邻点和已检测的满足检测准则的点合，从而产生小块目标区域，然后再检查该区域的全部邻点，并把满足跟踪准则的邻点并入这个目标区域，不断重复上述操作，直到没有邻点满足跟踪准则为止，则此块区域生长结束。然后用检测准则继续寻找，当找到满足检测准则且不属于任何已生成的区域的像素点后，开始下一个区域的生长，按这种方法进行，直到没有满足检测准则的像素点为止。

MATLAB 提供了两个边界跟踪函数：bwtraceboundary()函数和 bwboundaries()函数，其功能见表 8-2。

表 8-2　边界跟踪函数介绍

函数名称	用　法
bwtraceboundary	采用基于曲线跟踪的策略，需要给定搜索起始点和搜索方向，返回过该起点的一条边界
bwboundaries	属于区域跟踪算法，能给出二值图像中所有物体的外边界和内边界

下面对它们的使用进行具体说明并给出实例。

1．Bwtraceboundary()函数

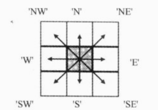
其语法格式如下：

- B=bwtraceboundary(BW, P, fstep)
- B=bwtraceboundary(BW, P, fstep, conn)
- B=bwtraceboundary(···, N, dir)

其参数的含义见表 8-3。

表 8-3　bwtraceboundary()函数中参数的含义

参数名称	含　义
BW	图像矩阵，值为 0 的元素视为背景像素点，非 0 元素视为待提取边界的物体
P	2×1 维矢量，两个元素分别对应起始点的行坐标和列坐标
fstep	字符串，指定起始搜索方向，它的取值与方向的对应关系如图 8-23 所示，见表 8-4
conn	指定搜索算法所使用的连通方法，具体参数设置见表 8-4
N	指定提取的边界的最大长度，即这段边界所含的像素点最大数目
dir	字符串指定搜索边界的方向，具体参数设置见表 8-4



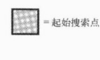


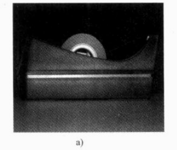
图 8-23 fstep 的取值和含义

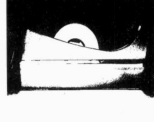
表 8-4 dir、conn、fstep 参数设置

参数		描述
dir	'clockwise'	在 clockwise 方向搜索（默认值）
	'counterclockwise'	在 counterclockwise 方向搜索
conn	4	4 连通（上、下、左、右）
	8	8 连通（上、下、左、右、右上、右下、左上、左下）
fstep	'N'	从图像上方开始搜索
	'S'	从图像下文开始搜索
	'E'	从图像右方开始搜索
	'W'	从图像左方开始搜索
	'NE'	从图像右上方开始搜索
	'SE'	从图像右下方开始搜索
	'NW'	从图像左上方开始搜索
	'SW'	从图像左下方开始搜索

输出中的 ***B*** 为一个 $Q\times2$ 维矩阵，其中 Q 为所提取的边界长度（即边界所含像素点的数目），***B*** 矩阵中存储边界像素点的行坐标和列坐标。

使用 bwtraceboundary()函数对 tape.png 图像进行边界提取，如图 8-24 所示。


a)


b)

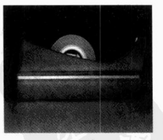

c)

图 8-24 bwtraceboundary()函数的应用

a) 原始图像 b) 二值图像 c) 应用结果

程序代码如下：

```
clear all
RGB=imread('tape.png');
```

```
figure,imshow(RGB)
I=rgb2gray(RGB);                    %将彩色图像转换成灰度图像
threshold=graythresh(I);            %计算将灰度图像转换成二值图像所需的门限
BW=im2bw(I,threshold);              %将灰度图像转换为二值图像
figure,imshow(BW)
dim=size(BW);
col=round(dim(2)/2)-90;             %计算起始列坐标
row=find(BW(:,col),1);              %计算起始点行坐标
connectivity=8;
num_points=180;
contour=bwtraceboundary(BW,[row,col],'N',connectivity,num_points);      %提取边界
figure,imshow(RGB);
hold on;
plot(contour(:,2),contour(:,1),'g','LineWidth',2);
```

2. bwboundaries()函数

其语法格式如下：

- B=bwboundaries(BW)
- B= bwboundaries(BW, conn)
- B= bwboundaries(BW, conn, options)
- [B, L]= bwboundaries(…)
- [B, L, N, A]= bwboundaries(…)

其中，BW、conn 与表 8-3 中意义相同，其他参数的含义见表 8-5。

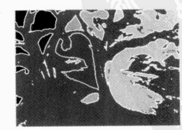
表 8-5 bwboundaries()函数中参数的含义

参数名称	含义
options	为字符串，取值为'holes'或'noholes'，其中前者为默认值，它指定算法既搜索物体的外边界、也搜索物体的内边界（即洞的边界），后者使算法只搜索物体的外边界
L	标志了该图像被边界所划分的区域，包括物体和洞。它是一个整数矩阵，与原始图像具有相同的维数，元素值代表了该位置上的像素点所在区域的编号，属于同一个区域的像素点对应的元素值相同
N	为该图像被边界所划分成的区域的数目，因此 N=max(L(:))
A	标志了被划分的区域的邻接关系，它是一个 $N \times N$ 维逻辑矩阵，其中 N 是被划分区域的数目，$A(i,j)$ =1 说明第 i 个区域与第 j 个区域存在邻接关系，且第 i 个区域(子区域)在第 j 个区域(父区域)内

使用 bwboundaries()函数对 trees.tif 图像中不同的区域标志不同的颜色，如图 8-25 所示。



a)



b)

图 8-25 bwboundaries()函数的应用

a) 原始图像 b) 标志区域





```
clear all
I=imread('trees.tif');
figure,imshow(I)
BW=im2bw(I,graythresh(I));                          %生成二值图像
%提取边界，并返回边界元胞数组 B 和区标志数组 L
[B,L]=bwboundaries(BW,'noholes');
figure,imshow(label2rgb(L,@jet,[.5 .5 .5]))         %以不同颜色标志不同的区域
hold on
for k=1:length(B)
    boundary=B{k};
    plot(boundary(:,2),boundary(:,1),'W','LineWidth',2)   %在图像上叠画边界
end
```

8.4.2 直线提取算法

由于直线具有不同于一般曲线的特征，因此它的提取方法也与一般的边界检测方法不同。

1. Hough 变换的基本原理

Hough 变换是最常用的直线提取方法，它的基本思想是：将直线上每一个数据点变换为参数平面中的一条直线或曲线，利用共线的数据点对应的参数曲线相交于参数空间中一点的关系，使直线的提取问题转化为计数问题。Hough 变换提取直线的主要优点是受直线中的间隙和噪声影响较小。

具体地说，对于满足直线方程 $y = ax + b$ 的某一个数据点 (x_0, y_0)，对应参数平面 (a,b) 上的一条直线 $b = y_0 - ax_0$，而来自于同一条直线 $y = a_0x + b$ 上的所有数据点对应的参数平面上的直线必然相交于真实的参数点 (a_0, b_0)。另外，为了避免垂直直线斜率无穷大的问题，在应用时通常采用直线的极坐标方程 $\rho = x\cos\theta + y\sin\theta$，此时参数平面为 (ρ,θ) 平面。图 8-26 给出了 Hough 变换基本原理的示意图。

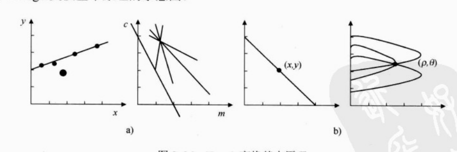
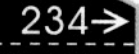


图 8-26 Hough 变换基本原理

a) (x,y)空间到(a,b)空间的交换　b) (x,y)空间到(ρ,θ)空间的交换

在算法实现中，考虑到噪声的影响和参数空间离散化的需要，求交点的问题成为一个累加器问题。算法步骤如下：

1）适当地量化参数空间。



2）假定参数空间的每一个单元都是一个累加器。

3）累加器初始化为零。

4）对图像空间的每一点，在其所满足参数方程对应的累加器上加 1。

5）累加器陈列的最大值对应模型的参数。



2. Hough 变换的 MATLAB 实现

MATLAB 提供了 3 个与 Hough 变换有关的函数：Hough()函数，Houghpeaks()函数和 Houghlines()函数，其作用见表 8-6。

表 8-6 Hough 变换函数

函数名称	用法
Hough	对图像进行 Hough 变换
Houghpeaks	用来提取 Hough 变换后参数平面上的峰值点
Houghlines	用于提取参数平面上的峰值点对应的直线

（1）Hough()函数

Hough()函数对输入图像进行标准 Hough 变换，采用极坐标方程 $\rho = x\cos\theta + y\sin\theta$ 中的直线参数 (ρ,θ) 作为参数平面。

其语法格式如下：

- [H, theta, rho]=hough(BW)
- [H, theta, rho]= hough(BW, param1, val1, param2, val2)

其中，BW 为输入的二值图像。param1，val1 和 param2，val1 共同制定参数平面的离散度。Hough()函数中参数的含义见表 8-7。

表 8-7 Hough()函数中参数的含义

参数	描述	
param	'ThetaResolution'	val 为θ轴的单元大小，它是介于 0~0° 的标量
	'RhoResolution'	val 为ρ轴的单元大小，它是介于 0~norm(size(BW)) (即以像素点数目衡量图像对角线的长度)的标量

Hough()函数输出中的 $\boldsymbol{H}$ 为参数平面的计数结果矩阵。维数为 $N_\rho \times N_\theta$，其中，N_ρ 为 ρ 轴离散后的单元数目，N_θ 为 θ 轴离散后的单元数目。$\boldsymbol{H}$ 矩阵中的元素值是参数平面上对应单元的计算结果。theta 为 $N_\theta \times 1$ 维矢量，指示 θ 轴各个单元对应的 θ 值。rho 为 $N_\rho \times 1$ 维矢量，指示 ρ 轴各个单元对应的 ρ 值。

（2）Houghpeaks()函数

Houghpeaks()函数用于提取 Hough 变换后参数平面上的峰值点。

其语法格式如下：

- Peaks=houghpeaks(H, numpeaks)
- Peaks=houghpeaks(…, param1,val1,param2,val2)

其参数的含义见表 8-8。

param 和 val 参数对用来指定寻找峰值的门限或峰值对周围像素点的抑制范围。具体取值及意义见表 8-9。由于噪声的影响，一个真实的参数点很可能和它周围的点同时超过峰值门限，而实际上只需在这个小区域内提取出一个峰值点，因此在提取出一个极大值点后，算法将它的领域内的计数器都置为 0，这个领域即由[*M*, *N*]指定的抑制区，[*M*, *N*]的默认值是大于或等于 size(H)/50 的最小奇数对。

表 8-8 Houghpeaks()函数中参数的含义

参数名称	含　义
H	Hough()函数的输出，参数平面的计数结果矩阵
numpeaks	指定要提取的峰值数目，默认值为 1
N	为该图像被边界所划分成的区域的数目，因此 N=max(L(:))
Peaks	$Q\times 2$ 维矩阵，其中 Q 为提取的峰值数目，Peaks 的第 q 行分别存储第 q 个峰值的行坐标和列坐标

表 8-9 Houghpeaks()函数中参数的含义

参　数		描　述
param	'Threshold'	val 参数指定峰值门限，可以是任意正实数，默认值为 0.5×max(H(:))
	'NHoodSize'	val=[*M*, *N*]，其中两个元素 *M* 和 *N* 都是正奇数，*M* 和 *N* 共同指定峰值周围抑制区的大小

（3）Houghlines()函数

Houghlines()函数根据 Hough 变换的结果提取图像中的直线。

其语法格式如下：

- Lines=houghlines(BW, theta, rho, peaks)
- Lines=houghlines(⋯,param1, val1, param2, val2)

其中，BW 与 Hough()函数的输入 BW 相同，为二值图像；theta 和 rho 为 Hough()函数返回的输出，指示θ轴和ρ轴各个单元对应的值。peaks 为 Houghpeaks()函数返回的输出，指示峰值的行坐标和列坐标，Houghlines()函数将根据这些峰值提取直线。param 和 val 是参数对，指定是否合并或保留直线段的相关函数。其参数的具体含义见表 8-10。

表 8-10 Houghlines()函数中参数的含义

参　数		描　述
param	'MinLength'	val 指定合并后的直线被保留的门限长度，长度小于 val 给定的门限值的直线被舍去，门限长度的默认值为 40
	'FillGap'	val 指定直线被合并的门限间隔，若两条斜率和截距均相同的直线段间隔小于 val 给定的值，则它们被合并为一条直线，门限间隔的默认值为 20

输出中的 Lines 为结构数组，数组长度为提取出的直线的数目。结构数组的每个元素存储一条直线段的相关信息，它包括的域见表 8-11。

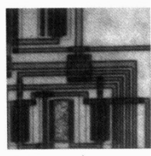
表 8-11 输出 Lines 中参数的意义

域 名	意 义
point1	二元矢量[x, y]指定直线段一个端点的行坐标和列坐标
point2	二元矢量[x, y]指定直线段另一个端点的行坐标和列坐标
thetaAngle	该线段对应的θ值，单位为度
rho	该线段对应的ρ值

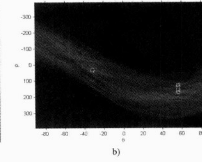


3．MATLAB 实用

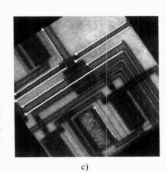
寻找图像中的直线段和其中最长的段，如图 8-27 所示。



a)



b)



c)

图 8-27 Hough 检测直线的应用

a) 原始图像 b) Hough 变换 c) 画直线并计算长度

程序代码如下：

```
clc;
clear all;
I=imread('circuit.tif');
figure,imshow(I)
rotI=imrotate(I,33,'crop');
BW=edge(rotI,'canny');
[H,T,R]=hough(BW);
figure,imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');
xlabel('\theta'),ylabel('\rho');
axis on, axis normal,hold on;
P=houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
x=T(P(:,2));y=R(P(:,1));
plot(x,y,'s','color','white');
lines=houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
figure,imshow(rotI),
hold on;
max_len=0;
for k=1:length(lines)
    xy=[lines(k).point1;lines(k).point2];
    plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
```

```
        plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
        plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
        len=norm(lines(k).point1-lines(k).point2);
        if(len>max_len)
            max_len=len;
            xy_long=xy;
        end
end
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan')
```

8.5 基于图像分割的图像分析

MATLAB 7.0 图像处理工具箱提供了几个基于图像分割的分析示例，通过对这几个示例的分析与理解，可以加深对图像分割的理解，同时也能加深对 MATLAB 7.0 提供的图像分割相关函数的熟悉，为以后的图像分析处理奠定良好的基础。这里，将以基于图像分割细胞检测和图像粒度测定为例进行说明。

8.5.1 通过图像分割检测细胞

细胞检测是生物学和医学研究中最基本的步骤，而通过图像分割来检测是现在常用的最直接的计算检测方法。

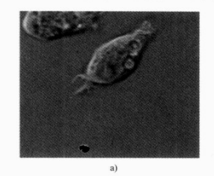
整个检测过程主要包括以下几个部分：

（1）图像读入

```
I=imread('cell.tif');
figure,imshow(I);
```

（2）检测完整的细胞

如图 8-28a 所示，图像中存在两个细胞，只有一个完整的，需要将这个完整的细胞分割出来。由于目标图像与背景有较大的区别，可以利用灰度的梯度信息来实现图像分割。为此，采用 Sobel 算子来实现边缘提取，边缘提取效果如图 8-28b 所示。


a)

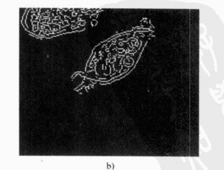

b)

图 8-28 原始图像及边缘提取结果

a) 原始图像 b) 边缘提取效果

```
BW=edge(I,'sobel',(graythresh(I)*.1));
figure,imshow(BW)
```

（3）填补缝隙

由检测效果图 8-28b 可以看到，虽然 edge()函数提取了图像的大概轮廓，但是边缘线存在断裂的情况，没有完整而精确地描绘出细胞的轮廓，在这里，可以通过 strel()函数利用线性的结果函数对边缘进行膨胀操作，填补边缘缝隙。

```
se90=strel('line',3,90);
se0=strel('line',3,0);
```

（4）膨胀操作

用 imdilate()函数对图像进行膨胀操作，膨胀结果如图 8-29a 所示。

```
BWsid1=imdilate(BW,[se90,se0]);
figure,imshow(BWsid1),
```

（5）填充

膨胀后的灰度图像精确地显示了细胞的外围轮廓，但是在细胞内部还有一些孔隙。可以利用 imfill()函数对这些孔隙进行填充，填充结果如图 8-29b 所示。

```
BWdfill=imfill(BWsid1,'holes');
figure,imshow(BWdfill)
```


a)

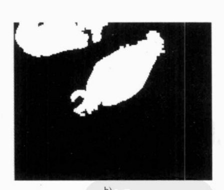

b)

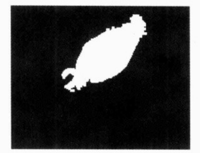
图 8-29 膨胀与填充操作结果

a) 膨胀操作 b) 填充操作

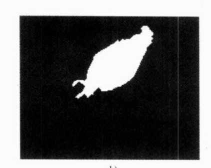
（6）移除与边界连通的目标

至此，可以对感兴趣的细胞进行成功分割，但是画面上还有其他物体，可以通过 imclearborder()函数来清除与边界连通的物体，得到如图 8-30 所示的分割结果。

```
BWbord=imclearborder(BWdfill,4);
figure,imshow(BWbord)
```



a)



b)

图 8-30　分割结果

a) 分割　b) 平滑

（7）平滑

对于分割的结果，边缘不是很光滑，需要利用菱形结构元素对图像进行平滑处理。

```
seD=strel('diamond',1);
BWfinal=imerode(BWbord,seD);
BWfinal=imerode(BWfinal,seD);
figure,imshow(BWfinal)
```

然后在原图上以轮廓线标出细胞的轮廓，至此，细胞的检测就完成了，如图 8-31 所示为标示结果。

程序代码如下：

```
BWoutline=bwperim(BWfinal);
Segout=I;
Segout(BWoutline)=255;
figure,imshow(Segout)
```

8.5.2 图像粒度测定

此应用示例是检测如图 8-32 所示图像中米粒的分布情况，即统计图像中米粒的大小及其所占的比例。

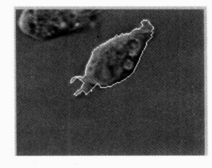


图 8-31　分割结果标记

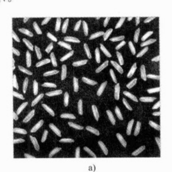


图 8-32　rice 图像

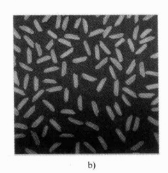
1）读入图像。

```
I=imread('rice.png');
figure,imshow(I)
```

2）将图像数据转换为双精度类型进行计算，这样才能充公发挥 MATLAB 语言进行向量处理的特长，并将图像的精确度范围调节为[0 1]，即设置为全部灰度范围。调整后图像如图 8-33 所示。



a)



b)

图 8-33 调整数据类型

a) imadjust 运算结果 b) 转换为双精度及灰度范围

```
claheI=adapthisteq(I,'NumTiles',[10 10]);
claheI=imadjust(claheI);
figure,imshow(claheI);
gI=imadjust(im2double(I),[],[0 1]);
figure,imshow(gI)
```

3）从图 8-33 可以看出，由于某些米粒位于背景较亮的地方，因而显得很模糊，不利于直接进行检测，为此，利用一个高帽变换来消除图像背景中那些不一致的背景亮度，去除效果如图 8-34 所示。

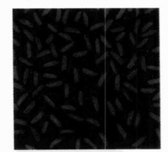


图 8-34 背景去除

```
se=strel('disk',10); %确定形态操作结构元素
topI=imtophat(gI,se); %高帽变换
figure,imshow(topI)
```

4）这样，就可以根据变换后的图像（见图 8-34），分析原始图像中米粒的大小分布情况了。这里利用一个逐渐变大的结构元素，不断对变换图像进行形态开启操作，并统计开启后对象的剩余面积，通过绘制结构元素的大小和剩余面积的大小就可以计算出各种大小的米粒在图像中占有的比例。米粒大小分布情况如图 8-35 所示。

```
I=imread('rice.png');
claheI=adapthisteq(I,'NumTiles',[10 10]);
claheI=imadjust(claheI);
gI=imadjust(im2double(I),[],[0 1]);
```

```
se=strel('disk',10);
topI=imtophat(gI,se);
for counter=0:22
    remain=imopen(claheI,strel('disk',counter));
    intensity_area(counter+1)=sum(remain(:));
end
figure,plot(intensity_area,'m-*'),grid on;
xlabel('radius of opening(pixels)');
ylabel('pixel value sum of opened objects(intensity)');

for counter=0:20 %利用一个逐渐变大的结构元素对变换图像进行形态开启操作
    remain=imopen(topI,strel('disk',counter));
    surfarea(counter+1)=sum(remain(:));
end
figure,plot(surfarea,'m-*'),grid on;
set(gca,'xtick',[0 2 4 6 8 10 12 14 16 18 20]);
xlabel('radius of opening(pixels)');
ylabel('surface area of opened objects(pixels)');
```

执行程序后得到如图 8-35 所示的效果。

5）从图 8-35 所示的曲线可以看出，随着结构元素的增大，对象的剩余面积发生锐减，这是由于原始图像中含有较多的相同大小星体的缘故。通过计算两次形态开启操作前、后的斜率（即一阶偏导）就可以估计出图像中相同大小米粒所占的比例，结果如图 8-36 所示。横轴为米粒半径大小，纵轴表示两次形态开启操作前、后的斜率。

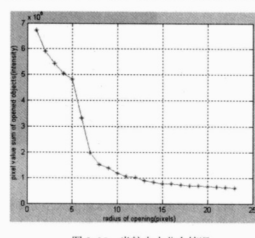


图 8-35 米粒大小分布情况

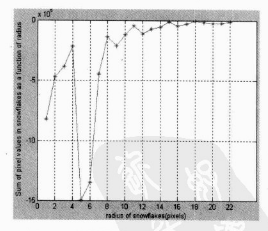


图 8-36 估计图像中相同大小米粒所占的比例

```
intensity_area_prime=diff(intensity_area);
plot(intensity_area_prime,'m-*'),grid on;
set(gca,'xtick',[0 2 4 6 8 10 12 14 16 18 20 22]);
xlabel('radius of snowflakes(pixels)');
ylabel('Sum of pixel values in snowflakes as a function of radius');
```

8.6 彩色图像分割



彩色图像分割是数字图像处理领域中一类非常重要的图像分析技术，在对图像的研究和应用中，根据不同领域的不同需要，在某一领域往往仅对原始图像中的某些部分感兴趣。这些目标区域一般来说都具备自身特定的一些诸如颜色、纹理等性质，彩色图像的分割主要根据图像在各个区域的不同特性，而对其进行边界或区域上的分割，并从中提取出所关心的目标。

图像分割注重对图像中的目标进行检测与测量，这与在像素级对图像进行操作的图像处理技术，为改善图像视觉效果而强调在图像之间所进行的变换是有所区别的。通过对图像的分割、目标特征的提取，可将经初步图像处理的图像特征向量提取出来，并将原始的数字图像转化成为一种有利于目标表达的更抽象、更紧凑的表现形式，从而使高层的图像分析、图像理解以及计算机的模式自动识别成为可能。多年来，彩色图像分割技术在工业自动化控制、遥感遥测、微生物工程以及合成孔径雷达（SAR）成像等多种工程应用领域得到了广泛的应用。

8.6.1 色彩空间

表达颜色的色彩空间有许多种，它们是根据不同的应用目的而提出的。下面将围绕彩色图像分割，介绍几种常用的色彩空间和它们的特点。

最常见的色彩空间是红绿蓝（Red、Green、Blue，RGB）空间，它是一种矩形直角空间结构的模型，是通过对颜色进行加运算完成颜色综合的彩色系统。它用 R、G、B 三个基本分量的值来表示颜色，它是面向硬件设备的（如 CRT 显示器），物理意义明确但缺乏直观感。与它对应的是深蓝、品红和黄的 CMY 空间，主要用于非发射式显示，如彩色打印机、绘画等。

通过对不同类型图像的分析，有人经过大量试验提出可用由 R、G、B 经过线性变换得到的三个正交彩色特征来进行分割。

$$I_1 = (R+G+B)/3$$

$$I_2 = (R-B)/2 \text{或} I_2 = (B-R)/2$$

$$I_3 = (2G-R-B)/4$$

这三个特征中，I_1 是最佳特征，I_2 是次佳特征，只用 I_1 和 I_2 作为特征，对大多数图像已可得到较好的分割效果。

彩色图像常用 R、G、B 三分量的值来表示，但 R、G、B 三分量之间常有很高的相关性，直接利用这些分量常常不能得到所需的效果。为了降低彩色特征空间中各个特征分量之间的相关性，以及为了使所选的特征空间更方便于彩色图像分割方法的具体应用，实际中常需要将 RGB 图像变换到其他的色彩空间中去。

比较接近人对颜色视觉感知的是色度、饱和度、亮度（Hue、Saturation、Intensity，HSI）空间。其中，I 表示颜色的明暗程度，也有用 V（value）表示的，主要受光源强弱影响，H 表示不同颜色，如黄、红、绿，而 S 表示颜色的深浅。注意，HSI 模型有两个重要的事实作为基础，首先，I 分量与彩色信息无关；其次 H 分量和 S 分量与人感受彩色的方式紧

密相连。HSI 空间比较直观并且符合人的视觉特性，这些特点使 HSI 模型非常适合基于人的视觉系统对彩色感知特性的图像处理。从 RGB 到 HSI 的转换关系为

$$H = \arccos((2R - G - B)/(2\mathrm{sqrt}((R - G)^2 + (R - B)(G - B)))$$

$$B > G$$

$$H = 2Pi - H$$

$$S = 1 - 3(\min(R,G,B))/(R + G + B)$$

$$I = (R + G + B)/3$$

其中，S 也有用下式计算的：$S = \max(R,G,B) - \min(R,G,B)$

另外，孟塞尔色彩空间也用一个三维空间的模型将各种表面色的三种视觉特性：亮度、色度、饱和度全部表示出来。孟塞尔色彩空间的颜色样品在视觉上是均匀的，因而可以用它来考察和验证与某一色差公式有关的颜色空间的均匀性。孟塞尔色彩空间的色度值、亮度值、彩度值大致反映了物体颜色的规律，代表了颜色的色度、亮度和饱和度的主观特性，但孟塞尔色彩空间完全以主观色表为基础，没有数学表达式，使用起来很不方便。除了这些常用的色彩系统外，还有如 YIQ、YUV、YCBCR 等色彩系统。

8.6.2 彩色分割方法

彩色图像分割是图像处理中的一个主要问题，也是计算机视觉领域低层次视觉中的主要问题。

总的来说，彩色图像分割的方法可以分为基于像元、区域和边缘的分割，前两类利用的是相似性，基于边缘的分割利用的是不连续性。

1．基于像元的分割方法

基于像元的分割方法又可分为三类：直方图门限技术、色彩空间聚类法以及模糊聚类分割方法。其中，直方图门限技术是最常用的，由于图像门限处理的直观性和易于实现的性质，使它在彩色图像分割应用中处于中心地位。

（1）直方图门限技术

Tominaga 提出可将 RGB 色彩空间转换成 HVC 色彩空间或其他色彩空间，如 HSI，再分别求 H、V、C（或 H、S、I）的一维直方图，寻找最明显的峰值，一般是选定两个作为门限。Holla 将 RGB 色彩空间转换成 RG、YB、I，再将这 3 个通道用带通滤波器平滑，滤波器中心频率过滤这三种色彩特征的比率是 I:RG:YB=4:2:1，然后在二维直方图 RG-YB 中寻找峰值点和基点，从而将像素点分成两个区域。但是该方法会在图像中留下捕捉不到的部分，因此可以再考虑将其他的特征添进去，如亮度或者像素的局部相连性，这样可以增强分割效果。Stein 的方法是对 Holla 的方法的改进，算法中加入了领域的特征。当留下了一些没有被分配到的像素点时，就取它周围的 3×3 的模板，如果模板中有一个或者多个像素点被指派到区域 A 中，则该像素点也被指派到同样的区域 A 中去。如果该领域模板中的像素点也没被指派到任何区域或者被指派到了不同的区域中，那么该像素点仍然不被指派。这样的话可能还是有残留点，但是比率要少得多。R.Ohtander 的方法是比较经典的，它采用九个色彩特征：R、G、B、H、S、V、Y、I、Q，对这九个特征分别计算直方图，再选择最好的峰值作为门限。Ohta 等提出的方法和前面方法的不同点在于它将 RGB 色彩空间转换为另外定义的 I1、I2、I3 特征，再分别对它们进行直方图化，从三个一维直方图

上可以看到各自的峰值点，该算法给出的 I1、I2、I3 的表达式相当于动态 K-L 变换的结果，而且都是对 R、G、B 的线性变换，不存在奇异点，不同的图像对 I1、I2、I3 各自的峰值点分割的效果有差别，需要自动选取合适的门限。根据 I1、I2、I3 的直方图，有明显双峰的更适合该图像。



（2）色彩空间聚类法

该方法结合了直方图阈值选取技术。先将 RGB 色彩空间转换成 HLS 色彩空间（H、L、S 的表达式已给出），根据 L 的值将图像分为过亮区域和非过亮区域，在过亮区域里以 H 为主要特征，根据直方图取峰值进行分割，在非过亮区域里以 S 为主要特征，根据直方图取峰值进行分割，最后将分割的两副图像合并。Ferri 则是通过神经网络将像素分成几个区域，再利用编辑和压缩技术来减少分类的个数。该方法用的是 YUV 色彩空间，它把每个像素点 (i,j) 扩展成矢量 $F(i,j)=\{U(i,j),V(i,j),U(i+h,j),V(i+h,j),U(i-h,j),V(i-h,j),U(i,j+h),V(i,j+h),U(i,j-h),V(i,j-h)\}$。其中，$h$ 是期望分割的目标的大小。Lauterbach 是在 LUV 色彩空间中进行分割的，首先求二维 UV 直方图的最高点，这个最高点是通过计算累计直方图的值和一个领域窗的均值之差得到的。然后添加色彩匹配线（acl），这条线是通过两个聚类中心的一根直线。像素值在 UV 空间的那两个聚类中心之间的 acl 的欧氏距离决定了像素点被分派到哪两个类中间去。最后在两类中用最小距离准则选择一类。但是，该方法没有考虑亮度，所以在某些情况下不太适用。

（3）模糊聚类分割法

模糊聚类分割法基于阈值和模糊 C-均值聚类法，先粗糙地用标量空间分析的一维直方图分割。其具体步骤是：计算图像每一个色彩特征的直方图；标量分析直方图；定义合法的几个类 V_2，V_2，…，V_c；对属于类别 V_i（$1 \leq i \leq c$）的每一个像素点 p，用 i 标记 p；计算每一类 V_i 的重心；对没有被分类的像素值 $p(x,y)$，用模糊成员函数 U 计算，取最大的 $U(x,y)$（此时类别为 V_k），则将该像素 p 点分派到 V_k。

2. 基于区域的分割方法

由于彩色图像分割的目的是将图像划分为不同区域。基于像素的分割是通过以像素性质的分布为基础的阈值来进行的，如灰度级的值或颜色，在这一节里讨论的是以直接找寻区域为基础的分割方法，主要可分为区域生长和区域分离与合并两类技术。

（1）区域生长

区域生长是一种根据事前定义的准则将像素或子区域聚合成更大区域的过程。其基本的方法是以一组种子点开始，将与种子性质相似（如灰度级或颜色的特定范围）的相邻像素附加到生长区域的每个种子上。不同的方法，其相似性准则不一样，该准则的选择不仅取决于面对的问题，还取决于有效图像数据的类型。一些基本的有某种一致性的区域是事先给定的，然后用不同的方法加入周围的领域。用边界松弛法分割：给定一个阈值 $D_{\max}$，如果平均距离 $D(R)<D_{\max}$，则区域 R 是一致的（即是同一类）。如果 ρ 属于区域 R_a，且 ρ 与区域 R_b 相邻，并且将 ρ 从区域 R_a 移至 R_b 的距离减去 $D(R_a)+D(R_b)$后，还可以满足 $D(R)<D_{\max}$，那么就将 P 从区域 R_a 移至 R_b。用 2×2 的块，通过加入相邻的像素来增长区域：如果像素点的颜色值与矩心的颜色值的差值小于 $2D_{\max}$，则加入该像素。除此之外，还可以用地理分水岭的算法来分割图像，该算法要求事先知道种子点的相关性；还可以根据一些统计数据来进行区域增长。或者用快速混合分割方法：由六边形领域的均值代替该点的像素值（即平滑

作用），这种分割方法是基于一种分等级的六边形结构组织，在最低层的时候，用局部区域增长法，这样可以获得小尺寸的相连区域；再上一层的时候，刚刚那层的每一个小尺寸的相连区域被看为是一个像素处理，同样再取六边形均值，再进行区域增长，这样一步一步进行下去。

（2）区域分离与合并

前面讨论的区域生长过程是从一组种子点开始的，另一种可作为替换的方法是在开始时将图像分割为一系列任意不相交的区域，然后将它们进行聚合或拆分。还有一种方法是先将图像分成彩色区域和非彩色区域，然后用直方图门限法，进行 8×8 块的合并，使用的色彩空间是 HSI。

3．基于边缘的分割方法

边缘检测对彩色图像分割是一个重要的工具，其分割方法可以分为两大类：一类是局部边缘检测技术；另一类是全局边缘检测技术。边缘检测技术常用到 Sobel、LOG 算子。局部边缘检测技术只需要考虑像素点领域的信息来决定一个边缘点，常用模糊 C-均值聚类法。全局边缘检测技术必须考虑到全局最优化，一般来说，很多全局边缘检测技术是基于马尔可夫随机过程的不同应用。还有的算法是先用 Canny 算子检测出强度边缘，然后在提取出来的所有边缘里根据色调和饱和度门限逐步消除掉一些边缘。

4．其他方法

除了上述 3 类彩色图像分割方法之外，还有一些彩色分割的综合方法，如分步分割方法：第一步借助取阈值方法进行粗略分割将图像转化为若干个区域；第二步利用模糊 C-均值聚类法将第一步剩下的像素进一步分类。这种方法可看做是由粗到细进行的，先用阈值方法是为了减少运用模糊 C-均值聚类法所需的计算量。

测量空间聚类法是分割彩色图像常用的方法，彩色图像在各个空间均可看做由 3 个分量构成，所以分割彩色图像的一种方法是建立一个 3-D 直方图，它可用一个 3-D 数组表示。这个 3-D 数组中的每个元素代表图像中给定三个分量值的像素的个数。阈值分割的概念可以扩展为在 3-D 空间搜索像素的聚类，并根据聚类来分割图像。该方法的优点是：首先，将图像由图像空间转换到测量空间的变换常是多对一的变换，这样变换后数据量减少，易于计算；其次，尽管许多聚类方法本质上是递归或迭代的，大部分聚类方法可以产生比较光滑的区域边界且不易于受噪声和局部边缘变化的影响。

马尔可夫随机场方法也可用于多分辨率彩色图像分割，这是一种收敛于最大后验概率的松弛方法。第一步的粗分割使用尺度空间滤波器以获得聚类的全局信息；第二步利用多尺度马尔可夫随机场细化分割。

习题

8-1　什么是图像分割？常用的图像分割方法可以分为哪几种类型？

8-2　设计一个程序能够找出具有双峰直方图特性图像的最佳分割阈值。

8-3　应用 MATLAB 语言编写实例，对 Sobel、Prewitt、Roberts、LOG、Canny 方法的边缘检测性能进行比较。

8-4　常用的基于梯度的边缘检测算子有哪些？它们各有何特点？

8-5 边缘检测算子有哪些？它们各有什么优缺点？请编程实现它们。

8-6 LOG 算子的基本原理是什么？它具有哪些优点？

8-7 什么是 Hough 变换？试述采用 Hough 变换检测直线的原理，用 MATLAB 语言编写相应的程序。

8-8 应用 MATLAB 语言编写对一幅灰度图像进行边缘检测、二值化的程序（检测和二值化的方法可以根据实际图像进行选择）。





第 9 章 小波分析及其在 MATLAB 中的应用

小波分析是建立在泛函数分析、Fourier 分析、样条分析及调和分析基础上的新的分析处理工具。它又被称为多分辨率分析，在时域和频域同时具有良好的局部化特性，常被誉为信号分析的“数学显微镜”。近十多年来，小波分析的理论和方法在信号处理、语言分析、模式识别、数据压缩、图像处理、数字水印、量子物理等专业和领域得到了广泛的应用。

近些年，小波分析被广泛用于图像的压缩、降噪、平滑和融合等方面，在人脸识别、医学图像处理、机器人视觉、数字电视等领域受到人们越来越多的重视。基于二维小波分析进行图像处理具有坚实的理论基础，MATLAB 软件在小波工具箱中也提供了强大的图像处理功能，包括采用命令行和图形用户接口等。

9.1 小波变换基础

9.1.1 连续小波变换

小波是通过对基本小波进行尺度伸缩和位移得到的。基本小波是一个具有特殊性质的实值函数，其震荡快速衰减，且在数学上满足积分为零的条件，即

$$\int_{-\infty}^{+\infty}\psi(t)\mathrm{d}t=0 \tag{9-1}$$

其频谱满足条件

$$C_{\psi}=\int_{-\infty}^{+\infty}\frac{\left|\psi(s)\right|^{2}}{s}\mathrm{d}s<0 \tag{9-2}$$

即基本小波在频域也具有较好的衰减性质。

一维连续小波基函数是通过尺度因子和位移因子由基本小波产生的，即

$$\psi_{a,b}(x)=\frac{1}{\sqrt{a}}\psi_{a,b}\left(\frac{x-b}{a}\right) \tag{9-3}$$

一维连续小波变换也称为积分小波变换，定义为

$$W_{f}(a,b)=(f,\psi_{a,b}(x))=\int_{-\infty}^{+\infty}f(x)\psi_{a,b}(x)\mathrm{d}t=\frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty}f(x)\psi_{a,b}\left(\frac{x-b}{a}\right)\mathrm{d}x \tag{9-4}$$

其逆变换为

$$f(x)=\frac{1}{C_{\psi}}\int_{0}^{+\infty}\int_{-\infty}^{+\infty}W_{f}(a,b)\psi_{a,b}(x)\mathrm{d}b\frac{\mathrm{d}a}{a^{2}} \tag{9-5}$$

二维连续小波基函数定义为

$$\psi_{a,b_x,b_y}(x,y)=\frac{1}{|a|}\psi\left(\frac{x-b_x}{a},\frac{y-b_y}{a}\right) \tag{9-6}$$

二维连续小波变换是

$$W_f(a,b_x,b_y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(x,y)\ \psi_{a,b_x,b_y}(x,y)\mathrm{d}x\mathrm{d}y \tag{9-7}$$

二维连续小波逆变换为

$$f(x,y)=\frac{1}{C_\psi}\int_0^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}W_f(a,b_x,b_y)\ \psi_{a,b_x,b_y}(x,y)\ \mathrm{d}b_x\mathrm{d}b_y\frac{\mathrm{d}a}{a^3} \tag{9-8}$$

1. 滤波器族

这里将小波变换与一组带通线性（卷积）滤波器相联系，来解释小波变换的基本原理。首先定义尺度a上的一般小波基函数：

$$\psi_a(x)=\frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right) \tag{9-9}$$

这是用a作为尺度因子，并用$a^{-1/2}$规范了基本小波。若记其翻转共轭为

$$\tilde{\psi}_a(x)=\frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right)$$
$$\tilde{\psi}_a(x)=\psi_a^\phi(x)=\frac{1}{\sqrt{a}}\psi^\phi\left(-\frac{x}{a}\right) \tag{9-10}$$

小波变换就可以表示成滤波器族：

$$W_f(a,b)=\int_{-\infty}^{+\infty}f(x)\tilde{\psi}_a(b-x)dx=f*\tilde{\psi}_a \tag{9-11}$$

而且每个滤波器的输出分量再次滤波并适当伸缩后组合在一起可重构$f(x)$：

$$f(x)=\frac{1}{C_\psi}\int_0^{+\infty}\int_{-\infty}^{+\infty}(f*\tilde{\psi}_a)(b)\psi_a(b-x)\frac{\mathrm{d}a}{a^2}$$
$$=\frac{1}{C_\psi}\int_0^{+\infty}(f*\tilde{\psi}_a*\psi_a)(x)\frac{\mathrm{d}a}{a^2} \tag{9-12}$$

图9-1表示了对一个信号的小波变换滤波器族的分析。

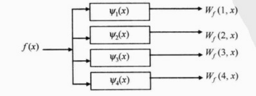


图9-1 对一个信号的小波变换滤波器族的分析

2. 二维滤波器族

在二维情况下，每一个滤波器都是一个二维冲激响应，输入是图像上的带通滤波器，滤

波后的图像的叠层组成了小波变换。图 9-2 对二维滤波器族进行了说明。

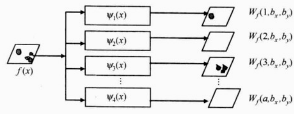


图 9-2 二维滤波器族示意图

9.1.2 离散小波

在数值计算中，需要对小波变换的尺度因子、位移因子进行离散化，一般采用如下的离散化方式：

令尺度因子 $a=a_0^m$， $b=na_0^m b_0$ （其中， $a>1$， $b\neq 0$， m， n 为整数）

小波基函数为

$$h_{m,n}(x)=\frac{1}{\sqrt{a_0^m}}h\left(\frac{1}{a_0^m}x-nb_0\right) \tag{9-13}$$

适当选择 h， a_0， b_0 使 $h_{m,n}(x)$ 构成规范正交基。

通常采用 $a_0=2$， $b_0=1$ 构成离散二进小波。

例如，在平方可积函数空间 $L^2(R)$ 中，最典型的规范正交基是 Haar 基，取 $a_0=2$， $b_0=1$ 时，小波函数族为

$$h_{m,n}(x)=\frac{1}{\sqrt{2^m}}h\left(\frac{1}{2^m}x-n\right) \tag{9-14}$$

1. 多分辨分析

基本小波通过伸缩构成一组基函数，在大尺度上，膨胀的基函数搜索大的特征；而在较小的尺度上，它们则寻找细节信息。

（1）金字塔算法

对于数字图像（以 512×512 为例），通过连续平均 2×2 的像素块并丢掉隔行隔列的像素，将得到缩小 4 倍的图像（256×256，行列各缩小 2 倍）。这样迭代进行，直到得到 1×1 的图像为止。如果利用同样尺寸的边缘检测算子（如 3×3 的 Sobel 算子），在原始图像上则会得到小边缘，在 256×256 及更小的图像上会得到稍大及更大的边缘。

（2）拉普拉斯金字塔编码

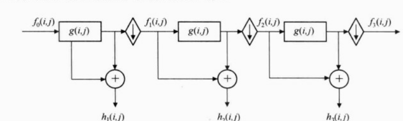
对原始图像 $f_0(i,j)(N\times N, N=2^n)$ 作高斯滤波，将图像分解为半分辨率的低频分量和整分辨率的高频分量。

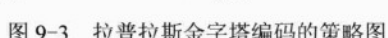
$$\begin{aligned} f_1(i,j)&=[f_0\times g](2i,2j) \\ h_1(i,j)&=f_0(i,j)[f_0\times g](i,j) \end{aligned} \tag{9-15}$$

这一过程是在间隔抽样后的图像上迭代进行的，经过 n 次迭代得到一组 $h_k(i,j)$ 和最终的低频图像 $f_n(i,j)$ （一个点）组成一个编码图像金字塔。图像的解码过程以相反的次序进行。从最后一幅图像 $f_n(i,j)$ 开始，对每一幅抽样图像 $f_k(i,j)$ 都进行一个增频采样，并与 $g(i,j)$ 卷

积进行内插。增频采样是在采样点之间插入零的过程，所得结果被添加到下一幅（前一幅）图像 $f_{k-1}(i,j)$ 上，再对所得图像重复执行这一过程。这个过程能无误差地重建出原始图像。

图 9-3 给出了拉普拉斯金字塔编码的策略图。




图 9-3　拉普拉斯金字塔编码的策略图

由于 $h_k(i,j)$ 图像在很大程度上降低了相关性和动态范围，因此可以使用较粗的量化等级，因而可以实现一个很大程度的图像压缩。

（3）子带编码和解码

对于有限带宽信号，若将其分解为窄带分量，当采用双通道子带时，对应带宽划分为两个分量（子带），如低半带和高半带，构造子带编码，这是一种时频域技术。低半带滤波器和高半带滤波器的建立如图 9-4 所示。

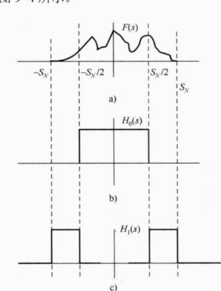


图 9-4　低半带滤波器和高半带滤波器的建立

a) 理想信号的频谱　b) 理想低半带滤波器　c) 理想高半带滤波器

双通道子带编码和解码，具有如下形式：

$$g_0(k\nabla t)=\sum_i f(i\nabla t)h_0((-i+2K)\nabla t) \tag{9-16}$$

$$g_1(k\nabla t)=\sum_i f(i\nabla t)h_1((-i+2K)\nabla t) \tag{9-17}$$

重建形式：

$$f(i\nabla t)=2\sum_{k}\left[g_0(k\nabla t)h_0((-i+2k)\nabla t)+g_1(k\nabla t)h_1((-i+2k)\nabla t)\right] \tag{9-18}$$

图 9-5 给出了双通道子带的编码和解码。

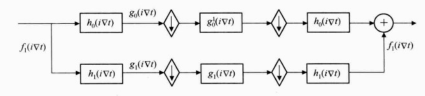


图 9-5　双通道子带的编码和解码

2．利用双通道子带编码迭代，自底向上建立小波变换

首先按照低半带和高半带进行子带编码后，对低半带再一次进行子带编码，得到一个 $N/2$ 点的高半带信号和对应于区间 $[0,s_N]$ 的第一个 1/4 区域和第二个 1/4 区域的两个 $N/4$ 点的子带信号。然后，连续进行上述过程，每一步都保留高半带信号并进一步编码低半带信号，直到得到一个仅有一个点的低半带信号为止。这样，小波变换系数就是这个低半带点再加上全部用子带编码的高半带信号，如图 9-6 所示。最前面的 $N/2$ 个系数来自于 $F(s)$ 的高半带，接下来的 $N/4$ 个点来自于第二个 1/4 区域，依次类推。

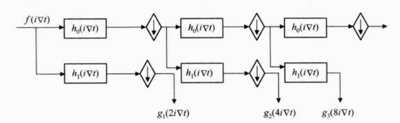


图 9-6　离散小波变换算法

图 9-6 给出了离散小波变换算法的处理流程。

上述算法被称为快速小波变换（Fast Wavelet Transform），也因其形状而被称为 Mallat 的“鱼骨型算法”。其逆变换算法的处理流程如图 9-7 所示。

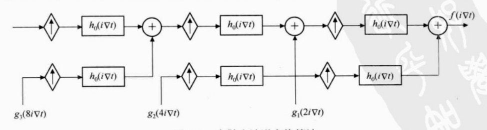


图 9-7　离散小波逆变换算法

3．离散小波变换的设计

根据上面介绍的子带编码重构公式，在频域上有

$$F(s)=2\left[\frac{1}{2}G_0(s)H_0(s)+\frac{1}{2}G_1(s)H_1(s)\right]$$
$$=F(s)H_0(s)H_0(s)+F(s)H_1(s)H_1(s) \quad (9\text{-}19)$$
$$=F(s)H_0(H_0^2(s)+H_1^2(s))$$

所以双通道子带编码的两个滤波器必须满足条件：

$$H_0^2(s)+H_1^2(s)=1,\quad 0\leqslant|s|\leqslant s_N \quad (9\text{-}20)$$

假设 $H_0(s)$ 是小波变换中使用的具有平滑边缘的低通滤波传递函数，则相应的 $H_1(s)$ 需按下式给出：

$$H_1^2(s)=1-H_0^2(s) \quad (9\text{-}21)$$

可见，设计一个离散小波变换的任务就是精心挑选低通滤波器。我们称符合这一条件的离散低通滤波器脉冲响应 $h_0(k)$ 为尺度向量，由它产生一个有关的函数称为尺度函数。尺度向量和尺度函数彼此互相确定。

例如，由尺度向量 $h_0(k)$ 到尺度函数的定义如下：

$$\Phi(t)=\sum_k h_0(k)\Phi(2t-k) \quad (9\text{-}22)$$

即它可以通过自身半尺度复制后的加权和来构造。另外它也能用带尺度的矩形脉冲函数卷积 $h_0(k)$ 利用数值计算方法得到：

$$\Phi(t)=\lim_{i\to\infty}\eta_i(x)$$
$$\eta_i(x)=\sqrt{2}\sum_n h_0(n)\eta_{i-1}(2x-n) \quad (9\text{-}23)$$

$$\eta_{oi}(x)=\prod(x)=\begin{cases}1 & |x|<1/2\\ 1/2 & |x|=1/2\\ 0 & |x|>1/2\end{cases} \quad (9\text{-}24)$$

相反，由尺度函数开始，在它满足单位平移下正交归一条件时，尺度向量的计算方法如下：

$$\langle\Phi(t-m),\Phi(t-n)\rangle=\delta_{m,n}$$
$$\Phi_{j,k}(t)=2^{j/2}\Phi(2^j t-k), j=0,1,\cdots,k=0,1,\cdots,2^{j-1} \quad (9\text{-}25)$$
$$h_0(k)=\langle\Phi_{1,0}(t),\Phi_{1,k}(t)\rangle$$

4．二维离散小波变换

为了将一维离散小波变换推广到二维，只考虑尺度函数是可分离的情况，即

$$\Phi(x,y)=\Phi(x)\Phi(y) \quad (9\text{-}26)$$

式中，$\Phi(x)$ 是一维尺度函数，其相应的小波是 $\psi(x)$。下列 3 个二维基本小波是建立二维小波变换的基础：

$$\psi^1(x,y)=\Phi(x)\psi(y),\psi^2(x,y)=\Phi(y)\psi(x),\psi^3(x,y)=\psi(x)\psi(y) \quad (9\text{-}27)$$

它构成二维平方可积函数空间 $L^2(R^2)$ 的正交归一基：

$$\psi_{j,m,n}^1(x,y)=2^j\psi^1(x-2^j m,y-2^j n)\quad j\geqslant 0,l=1,2,3,\quad j,l,m,n \text{ 都为整数} \quad (9\text{-}28)$$

（1）正变换

从一幅 $N\times N$ 的图像 $f(x,y)$ 开始，其中上标指示尺度，并且 N 是 2 的幂数。对于 $j=0$，尺度 $2^j=2^0=1$，也就是原始图像的尺度。j 值的每一次增大都使尺度加倍，而使分辨率减半。在变换的每一层次，图像都被分解为 4 个四分之一大小的图像，它们都是由原始图像与一个小波基图像的内积后，再经过在行和列方向进行 2 倍的间隔抽样而生成的。对于第一个层次（j=1），可写成

$$\begin{aligned}
f_2^0(m,n)&=\left\langle f_1(x,y),\Phi(x-2m,y-2n)\right\rangle\\
f_2^1(m,n)&=\left\langle f_1(x,y),\psi^1(x-2m,y-2n)\right\rangle\\
f_2^2(m,n)&=\left\langle f_1(x,y),\psi^2(x-2m,y-2n)\right\rangle\\
f_2^3(m,n)&=\left\langle f_1(x,y),\psi^3(x-2m,y-2n)\right\rangle
\end{aligned}\tag{9-29}$$

后续的层次（j>1），依次类推，形成如图 9-8 所示的形式。

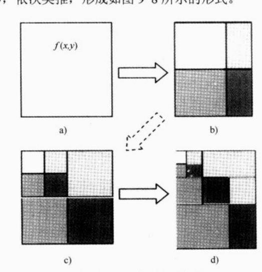


图 9-8　二维离散小波变换

a) 原始图像　b) 第一层　c) 第二层　d) 第三层

若将内积改写卷积形式，则有

$$\begin{aligned}
f_{2^{j+1}}^0(m,n)&=\left[f_{2^j}^0(x,y)*\Phi(-x,-y)\right](2m,2m)\\
f_{2^{j+1}}^1(m,n)&=\left[f_{2^j}^0(x,y)*\psi(-x,-y)\right](2m,2m)\\
f_{2^{j+1}}^2(m,n)&=\left[f_{2^j}^0(x,y)*\psi(-x,-y)\right](2m,2m)\\
f_{2^{j+1}}^3(m,n)&=\left[f_{2^j}^0(x,y)*\psi(-x,-y)\right](2m,2m)
\end{aligned}\tag{9-30}$$

因为尺度函数和小波函数都是可分离的，所以每个卷积都可分解成行和列的一维卷积。例如，在第一层，首先用 $h_0(-x)$ 和 $h_1(-x)$ 分别与图像 $f(x,y)$ 的每行作卷积并丢弃奇数列（以最左列为第 0 列）。接着这个 $(N\times N)/2$ 矩阵的每列再和 $h_0(-x)$、$h_1(-x)$ 作卷积，丢弃奇数行（以最上行为第 0 行）。结果就是该层变换所要求的 4 个 $(N/2)\times(N/2)$ 的数组，如图 9-9 所示。

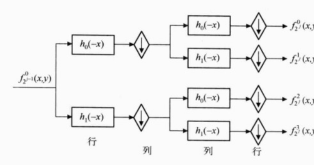


图 9-9 DWT 图像分解步骤



（2）逆变换

逆变换与上述过程相似，在每一层，通过在每一列的左边插入一列零来增频采样前一层的 4 个矩阵；接着用 $h_0(x)$ 和 $h_1(x)$ 来卷积各行，再成对地把这几个 $(N/2)\times N$ 的矩阵加起来；然后通过在每行上面插入一行 0 来将刚才所得的两个矩阵的增频采样为 $N\times N$；再用 $h_0(x)$ 和 $h_1(x)$ 与这两个矩阵的每列卷积。这两个矩阵的和就是这一层重建的结果。

图 9-10 给出了逆小波变换图像重建的过程。

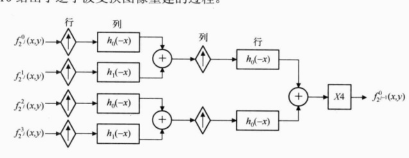


图 9-10 DWT 图像重建步骤

5. 双正交小波变换

使用两个不同的小波基，一个用来分解（分析），另一个用来重建（合成），构成彼此对偶的双正交的小波基：

$$\left\langle \psi_{j,k},\tilde{\psi}_{1,m}\right\rangle = \delta_{j,1}\delta_{k,m} \tag{9-31}$$

两个小波都能用于分解：

$$c_{j,k} = \left\langle f(x),\tilde{\psi}_{j,k}(x)\right\rangle \mathrm{d}_{j,k} = \left\langle f(x),\tilde{\psi}_{j,k}(x)\right\rangle \tag{9-32}$$

而重建为

$$f(x) = \sum_{j,k} \mathrm{c}_{j,k}\psi_{j,k}(x) = \sum_{j,k} \mathrm{d}_{j,k}\tilde{\psi}_{j,k}(x) \tag{9-33}$$

一维双正交小波变换通过 4 个离散滤波器实现，需要选择两个低通滤波器，即尺度向量，使它们的传递函数满足

$$H_0(0) = \tilde{H}_0(0) = 1 \text{且} H_0(s_N) = \tilde{H}_0(s_N) = 0 \tag{9-34}$$

式中，$s_N = \frac{1}{2}\Delta x$ 是折叠频率。

由它们产生两个带通滤波器（小波向量）：

$$
\begin{aligned}
h_1(n) &= (-1)^n h_0(1-n) \\
\tilde{h}_1(n) &= (-1)^n \tilde{h}_0(1-n)
\end{aligned}
\tag{9-35}
$$

双正交小波变换的一个分解步骤和一个重建步骤如图 9-11 所示。

图 9-11　双正交小波变换的一个分解步骤和一个重建步骤

$$
\begin{aligned}
\psi(x) &= \sqrt{2}\sum_n h_1(n+1)\Phi(2x-n) \\
\tilde{\psi}(x) &= \sqrt{2}\sum_n \tilde{h}_1(n+1)\Phi(2x-n)
\end{aligned}
\tag{9-36}
$$

正变换的二维基本小波为

$$\psi^1(x,y)=\Phi(x)\psi(y),\psi^2(x,y)=\Phi(y)\psi(x),\psi^3(x,y)=\psi(x)\psi(y) \tag{9-37}$$

逆变换的二维基本小波为

$$\tilde{\psi}^1(x,y)=\tilde{\Phi}(x)\tilde{\psi}(y),\tilde{\psi}^2(x,y)=\tilde{\Phi}(y)\tilde{\psi}(x),\tilde{\psi}^3(x,y)=\tilde{\psi}(x)\tilde{\psi}(y) \tag{9-38}$$

二维双正交小波变换就是由对应的小波基确定的。

9.1.3 二进小波变换

通常在数值计算中，采用离散化的尺度及位移因子，当取二进伸缩（以 2 作为因子伸缩）和二进位移（每次移动 $k/2^j$）时，就形成二进小波。

正交小波定义为满足下列条件的小波：

$$\psi_{j,\mathrm{k}}(x)=2^{j/2}\psi(2^j x-k),-\infty<j,k<+\infty,j,k \text{ 为整数} \tag{9-39}$$

它们构成$L^2(R^2)$（二维平方可积函数空间）中的正交归一基。j 决定伸缩，而 k 确定平移幅度。

正交性条件为

$$\langle \psi_{j,\mathrm{k}},\psi_{1,\mathrm{m}} \rangle = \delta_{jl}\delta_{km} \quad \text{（Kronecher 函数）} \tag{9-40}$$

任何 $f(x)\in L^2$ 都可展开为

$$f(x)=\sum_{-\infty}^{+\infty}\sum_{-\infty}^{+\infty} c_{j,\mathrm{k}}\psi_{j,\mathrm{k}}(x) \tag{9-41}$$

式中，变换系数为

$$c_{j,\mathrm{k}}=\langle f(x),\psi_{j,\mathrm{k}}(x)\rangle = 2^{j/2}\int_{-\infty}^{+\infty} f(x)\psi(2^j x-k)\mathrm{d}x \tag{9-42}$$

式（9-42）就是小波级数展开公式。

当进一步把 $f(x)$ 和基本小波限制为在[0,1]区间外为零的函数时，上述正交小波函数族就成为紧致二进小波函数族，它可以用单一的索引 n 来确定：

$$\psi_n(x)=2^{f/2}\psi(2^j x-k) \tag{9-43}$$

式中，j 和 k 是 n 的如下函数：

$n=2^j+k, j=0,1,\cdots,k=0,1,\cdots,2^{j-1}$，即 j 是满足 $2^j \leqslant n$ 的最大整数，而 $k=n-2^j$。而相应的逆变换为

$$f(x)=\sum_{n=0}^{+\infty} c_n \psi_n(x) \tag{9-44}$$

式中，假定 $\psi_0(x)=1$，变换系数为

$$c_n=\langle f(x),\psi_n(x)\rangle=2^{j/2}\int_{-\infty}^{+\infty} f(x)\psi(2^j x-k)\mathrm{d}x \tag{9-45}$$

$$\psi_{a,b_x,b_y}(x,y)\frac{1}{|a|}\psi\left(\frac{x-b_x}{a},\frac{y-b_y}{a}\right) \tag{9-46}$$

Haar 基本小波函数是定义在区间[0,1]上的函数，如图 9-12 所示。

图 9-13 给出了两个小波 $\psi_{1,0}(x)$ 和 $\psi_{1,1}(x)$。

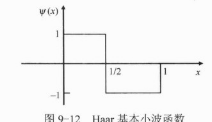


图 9-12 Haar 基本小波函数

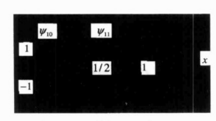


图 9-13 小波 $\psi_{1,0}(x)$ 和 $\psi_{1,1}(x)$

该基本小波定义的小波变换称为 Haar 小波变换，是常用的小波变换中最简单的一种。

9.1.4 MATLAB 中的小波函数工具箱

在 MATLAB 7.0 中有专门的小波函数工具箱，支持小波在图像处理中的应用。小波函数工具箱中的一维、二维小波应用函数见表 9-1 和表 9-2。

表 9-1 一维离散小波函数

函 数 名	功 能
appcoef	提取了一维小波分解低频系数
detcoef	提取一维小波分解高频系数
dwt	单层一维小波分解
dwtmode	离散小波变换扩展模式
idwt	单层一维逆离散小波变换
upcoef	一维小波分解的直接重构
upwiev	一维小波分解的单层重构
wavedec	多层一维小波分解
waverec	多层一维小波重构
wenergy	一维小波分解能量函数
wrcoef	二维小波分解系数单支重构



表 9-2　二维离散小波函数

函 数 名	功 能
dwt2	单层二维小波分解
dwtper2	单层二维离散小波变换
wavedec2	多层二维小波分解
idwt2	单层二维逆离散小波变换
idwper2	单层二维小波分解
waverec2	多层二维小波重构
upwiev2	二维小波分解的单层重构
wrcoef2	二维小波分解系数单支重构
upcoef2	二维小波分解的直接重构
detcoef2	提取二维小波分解高频系数
appcoef2	提取二维小波分解低频系数
wthresh	进行软阈值或硬阈值处理
wthcoef2	二维信号的小波系数阈值处理
ddencmp	获取在消噪或压缩过程中的默认阈值
wdencmp	用小波进行信号的消噪和压缩

9.2　小波分析在图像增强中的应用

前面已经详细讲述了图像增强技术，它是图像处理中最基本的技术之一，这里只介绍基于多层方法的增强技术。小波变换将一幅图像分解为大小、位置和方向均不相同的分量，在作逆变换之前，可根据需要对不同位置、不同方向上的某些分量改变其系数的大小，从而使得某些感兴趣的分量放大而使某些不需要的分量减小。其基本框图如图 9-14 所示。

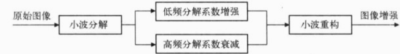
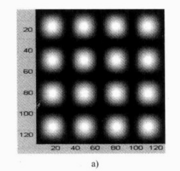
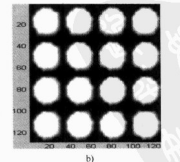


图 9-14　基于小波变换的图像增强基本原理

下面举例说明图像的增强。程序代码如下，执行结果如图 9-15 所示。



a)



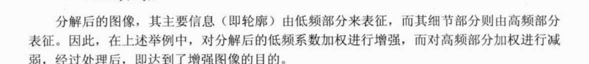
b)

图 9-15　基于小波的图像增强效果

a) sinsin 图像　b) 增强后的图像



```
%小波图像增强，对于图像，轮廓主要在低频，细节在高频
%对低频增强，对高频衰减处理
clear all
load sinsin;
subplot(1,2,1);image(X);
colormap(map);
%对图像 X 用小波 db5 进行 2 层分解
[c,l]=wavedec2(X,2,'db5');
Csize=size(c);
%弱化不重要的分解系数
for i =1:Csize(2)
    if(c(i)>100)
        c(i)=2*c(i);
    else
        c(i)=0.5*c(i);
    end
end
%重构图像并显示
X=waverec2(c,l,'db5');
subplot(1,2,2),image(X);
colormap(map);
```



分解后的图像，其主要信息（即轮廓）由低频部分来表征，而其细节部分则由高频部分表征。因此，在上述举例中，对分解后的低频系数加权进行增强，而对高频部分加权进行减弱，经过处理后，即达到了增强图像的目的。

9.3 基于小波的图像降噪和压缩

数字图像在产生过程中会受到诸如传感器振荡，电子器件干扰等原因的影响，导致转换后得到的数字图像质量下降，影响了对图像内容的理解。为了保证后续处理的正确性，需要对图像进行去噪处理。然而在图像去噪时，存在着一个如何兼顾降低噪声和保留细节的难题。长期以来，人们根据图像的特点、噪声的统计特征和频谱分布的规律，提出了多种去噪方法，如维纳滤波等，但是降噪效果往往不够理想。

小波变换具有低熵性、多分辨率、去相关性、选基灵活性等特点，可同时进行时域、频域的局部分析，能够灵活地对信号局部奇异特征进行提取。从信号学的角度看，小波去噪是一个信号滤波问题，小波去噪具有特征提取和低通滤波的综合功能。目前，基于小波分析的图像去噪技术已成为图像去噪的一个重要方法。

MATLAB 小波工具箱为图像降噪和压缩提供了有力的函数支持，见表 9-3。从本质上看，降噪和压缩的原理是基本类似的，都是对小波分解的系数进行阈值处理，然后进行重构。

表 9-3　基于小波的图像降噪和压缩函数

函数名称	说　明
wnoise	产生小波的噪声测试数据
ddencmp	获取降噪或压缩的默认值
wthresh	执行软阈值或硬阈值
thselect	选择降噪时的阈值
wbmpen	设置一维或二维信号降噪的阈值
wdcbm2	以 Birge-Massart 策略设置二维小波降噪或压缩的阈值
wthcoef2	二维小波系数阈值处理
wdencmp	用小波进行一维或二维信号的降噪或压缩
wthrmngr	阈值设置管理

另外，也可以利用小波包的一些函数进行压缩和降噪，见表 9-4。

表 9-4　小波包降噪和压缩函数

函数名称	说　明
ddencmp	获取降噪或压缩的默认值
wpbmpen	小波包降噪的 Penalized 阈值
wpthcoef	小波包系数阈值处理
wpdencmp	使用小波包进行降噪和压缩
wthrmngr	阈值设置管理

这里主要介绍一下其中降噪和压缩的核心函数：wdencmp()函数，其他就不再详细介绍了。读者可以参考一下工具箱的函数说明。wdencmp()函数的用法有 3 种：

- [XC,CXC,LXC,PERF0,PERFL2]= wdencmp('gbl',X, 'wname',N,THR,SORH,KEEPAPP)
- [XC,CXC,LXC,PERF0,PERFL2]= wdencmp('lvd',X, 'wname',N,THR,SORH)
- [XC,CXC,LXC,PERF0,PERFL2]= wdencmp('lvd',C,L, 'wname',N,THR,SORH)

函数中参数的含义见表 9-5。

表 9-5　wdencmp()函数中参数的含义

参数名称	含　义
[CXC,LXC]	附加的输出变量，是 XC 的小波分解结构
PERF0　PERFL	恢复和压缩的 L2 范数百分比
wname	小波函数名
N	小波分解的层数
SORH	取值为's'或者'h'，表示软阈值或硬阈值
[C,L]	小波分解结构
THR	3×N 矩阵，包含各尺度中水平、垂直和对角 3 个方向的阈值

9.3.1 小波的图像压缩技术

所谓图像压缩就是去掉各种冗余，保留重要的信息。图像压缩的过程常称为编码，而图像的恢复则称为解码。图像数据之所以能够进行压缩，其数学机理主要有下面两点。

1）原始图像数据往往存在各种信息的冗余（如空间冗余、视觉冗余和结构冗余等），数据之间存在相关性，邻近像素的灰度（将其看成随机变量）往往是高度相关的。

2）在多媒体应用领域中，人眼作为图像信息的接收端，其视觉对边缘的急剧变化敏感，以及人眼存在对图像的亮度信息敏感，而对颜色分辨率弱等，因此在高压缩比的情况下，解压缩后的图像信号仍有满意的主观质量。

虽然图像的数据是非常巨大的，但是可以采用适当的坐标变换去除相关，从而达到压缩数据的目的。传统的 K-L 变换就是以这种思想为基础的，它把信号的一小块看成是一个独立的随机向量，它的基函数由余弦函数组成。

小波变换通过多分辨分析过程将一幅图像分成近似和细节两部分，细节对应的是小尺度的瞬变，它在本尺度内很稳定。因此将细节存储起来，对近似部分在下一个尺度上进行分解，重复该过程即可，如图 9-16 所示。近似与细节在正交镜像滤波器算法中分别对应于高通滤波和低通滤波，这种变换通过尺度去掉相关性，在视频压缩中被证明是有效的。

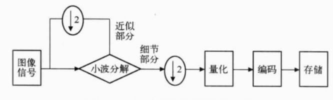


图 9-16　小波变换的图像压缩过程

上述方法有一个优点：即图像分成多个细节层，因此应用时可以先给出一幅较为粗糙的图像，然后可根据需要提供更好的细节。

应用 MATLAB 小波工具箱进行图像压缩，有两种方法。

1）利用 dwt2()函数对图像进行小波分解，再用 upcoef2()函数对分解后的图像进行重构，最后用 wcodemat()函数进行量化编码。程序代码如下：

```
I=imread('eight.tif')
figure(1),imshow(I);
%%%进行系数分解
[ca,ch,cv,cd]=dwt2(I,'sym4');
%%%进行图像重构
cod_fa=upcoef2('a',ca,'db1',1);
cod_fh=upcoef2('h',ch,'db1',1);
cod_fv=upcoef2('v',cv,'db1',1);
cod_fd=upcoef2('d',cd,'db1',1);
%%%编码显示图像
figure(2),imshow(wcodemat(cod_fa,255),[]);
figure(3),imshow(wcodemat(cod_fh,255),[]);
```



```
figure(4),imshow(wcodemat(cod_fv,255),[]);
figure(5),imshow(wcodemat(cod_fd,255),[]);
```

执行程序代码后效果如图 9-17 和图 9-18 所示。

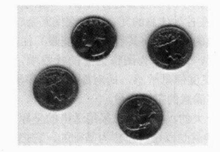
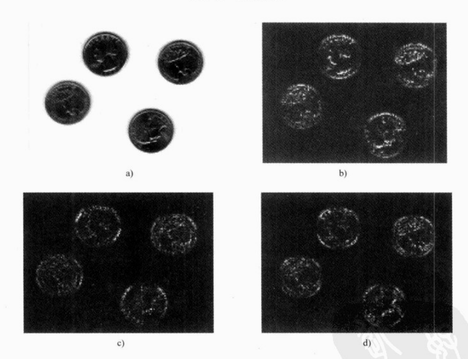


图 9-17　原始图像



a)　b)　c)　d)

图 9-18　小波分解

a) 近似值　b) 水平细节系数　c) 垂直细节系数　d) 对角细节系数

小波变换将图像的像素解相关的变换系数进行编码，比对原像素本身编码的效率更高。如果变换的基函数（此时为小波函数）将大多数重要的可视信息压缩到少量的系数中，（见图 9-18a），则剩下的系数（见图 9-18b～d）可以被粗略地量化或截取为 0，而图像几乎没有失真（比较图 9-17 与图 9-18a）。

2）利用小波工具箱中专用的阈值压缩图像函数 wdencmp()。

下面通过举例来说明这两种方法的应用及特点。

如下是利用 wavedec2()函数对图像进行小波分解后，再用 appcoef2()函数对分解后图像进行重构，最后用 wcodemat()函数进行量化编码，程序代码如下：

```
load belmont2
subplot(2,2,1);
image(X);colormap(map);axis square;
whos('X');
%对图像进行 7 层小波分解
[c,l]=wavedec2(X,2,'bior3.7');
%提取小波分解结构中的一层的低频系数和高频系数
cA1=appcoef2(c,l,'bior3.7',1);
%水平方向
cHl=detcoef2('h',c,l,1);
%斜线方向
cDl=detcoef2('d',c,l,1);
%垂直方向
cVl=detcoef2('v',c,l,1);
%重构第一层系数
A1=wrcoef2('a',c,l,'bior3.7',1);
H1=wrcoef2('h',c,l,'bior3.7',1);
D1=wrcoef2('d',c,l,'bior3.7',1);
V1=wrcoef2('v',c,l,'bior3.7',1);
c1=[A1,H1;V1,D1];
%显示第一层频率信息
subplot(2,2,2);image(c1);
%对图像进行压缩:保留第一层低频信息并对其进行量化编码
ca1=wcodemat(cA1,440,'mat',0);
%改变图像高度并显示
ca1=0.5*ca1;
subplot(2,2,3);image(ca1);colormap(map);
whos('ca1')
cA2=appcoef2(c,l,'bior3.7',2);
ca2=wcodemat(cA2,440,'mat',0);
ca2=0.5*ca2;
subplot(2,2,4);image(ca2),colormap(map);
whos('ca2')
```

结果如下：

压缩前图像的大小为

```
Name      Size                Bytes   Class
X         240×320             614400  double array
Grand total is 76800 elements using 614400 bytes
```

第一次压缩后图像的大小为

```
Name        Size                 Bytes   Class
ca1         127×167              169672  double array
Grand total is 21209 elements using 169672 bytes
```

第二次压缩后图像的大小为

```
Name        Size                 Bytes   Class
ca2         71×91                51688   double array

Grand total is 6461 elements using 51688 bytes
```

压缩后图像如图 9-19 所示。

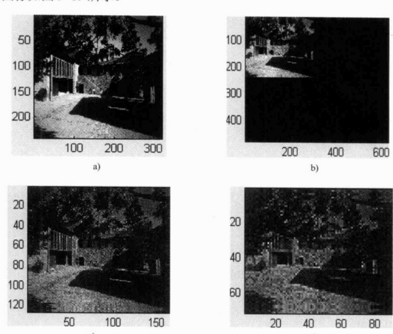


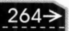
图 9-19　小波的图像压缩效果

a) 原始图像　b) 分解后的低频信息和高频信息　c) 第一次压缩后的图像　d) 第二次压缩后的图像

在这里可以看出，第一次压缩提取原始图像中小波分解第一层的低频信息，此时压缩效果较好，压缩比较小（约为 1/4 大小）。第二次压缩是提取第一层分解低频部分的低频部分（即第二层的低频部分），其压缩比较大（约为 1/2），压缩效果在视觉上也基本过得去。随着分解层数的增强，压缩比是递减的。

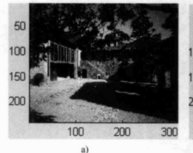
保留原始图像中低频信息的压缩办法只是一种最简单的压缩办法。它不需经过其他处理即可获得较好的压缩效果。当然，对于上面的例子还可以只提取小波分解的第三、第四层的低频信息。从理论上说，可以获得任意压缩比的压缩图像。只不过在对压缩比和图像质量都有较高的要求时，它就不如其他编码方法了。

如下是利用 wdencmp()函数对给定图像进行压缩处理。程序代码如下：

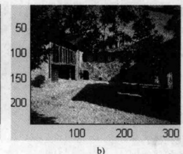


```
load belmont2;
subplot(2,2,1);
image(X);colormap(map);
%首先利用 db3 小波对图像 X 进行 2 层分解
[c,l]=wavedec2(X,2,'db3');
%全局阈值
[thr,sorh,keepapp]=ddencmp('cmp','wv',X);
%压缩处理:对所有高频系数进行同样的阈值量化处理
[Xcmp,cxc,lxc,perf0,perfl2]=wdencmp('gbl',c,l,'db3',2,thr,sorh,keepapp);
%将压缩后的图像与原始图像相比较
subplot(2,2,2);
image(Xcmp);colormap(map);
%显示相关参数
disp('小波分解系数中为 0 的系数个数百分比:');
perf0
disp('压缩后保留能量百分比:');
perfl2
```

首先给出应用函数 wdencmp()进行压缩的效率，即分解系数中置 0 的系数个数百分比和保留能量百分比。压缩图像如图 9-20 所示。



a)



b)

图 9-20 图像压缩结果

a) 原始图像 b) 压缩后的图像



结果如下：

小波分解系数中为 0 的系数个数百分比：

```
perf0 =
    40.1967
```

压缩后保留能量百分比：

```
perfl2 =
    99.9616
```

9.3.2 小波的图像降噪技术

根据对小波系数处理方式的不同，常见的去噪方法可分为三类：基于小波变换模极大值

去噪；基于相邻尺度小波系数相关性去噪；基于小波变换域阈值去噪。其中小波阈值去噪方法是一种实现简单且效果较好的消噪方法，应用最为广泛。

小波阈值消噪的基本思想就是对小波分解后的各层系数模大于和小于某阈值的系数分别进行处理，然后利用处理后的小波系数重构出消噪后的图像。在阈值消噪中，阈值函数体现了对小波分解系数的不同处理策略及不同估计方法，常用的阈值函数有硬阈值函数和软阈值函数。硬阈值函数可以很好地保留图像边缘等局部特征，但图像会出现伪吉布斯效应等视觉失真现象；而软阈值处理相对较平滑，但可能会造成边缘模糊等失真现象，为此人们又提出了半软阈值函数。

小波阈值消噪方法处理阈值的选取，另一个关键因素是阈值的具体估计。如果阈值太小，消噪后的图像仍然存在噪声；相反如果阈值太大，重要图像特征又将被滤掉，引起偏差。从直观上讲，对于给定的小波系数，噪声越大，阈值越大。

图像信号的小波降噪步骤有三步，和一维信号的降噪步骤完全相同，所不同的是，处理工具是用二维小波分析工具代替了一维小波分析工具。因此，对二维图像信号的降噪方法也同样适合于一维信号，尤其对几何图形更为适合。二维小波分析用于图像降噪的步骤如下：

1）二维图像信号的小波分解。在这一步，应当选择合适的小波和恰当的分解层次（记为 N），然后对待分析的二维图像信号进行 N 层分解计算。

2）对分解后的高频系数进行阈值量化。对于分解的每一层，选择一个恰当的阈值，并对该层高频系数进行软阈值量化处理。在此，阈值选取规则同前面的信号处理部分。

3）二维小波的重构图像信号。同样地，根据小波分解后的第 N 层近似（低频系数）和经过阈值量化处理后的各层细节（高频系数），来计算二维信号的小波重构。

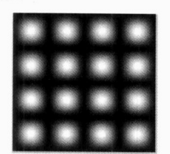
下面通过具体的举例来说明利用小波分析进行图像降噪这个问题。

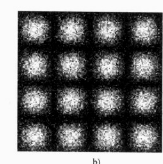
程序代码如下：

```
%当前延拓模式是补零
%载入原始图像
load sinsin
%X 包含原始图像
figure;
image(X),colormap(map);
%产生噪声图像
init=2055615866;randn('seed',init);
x=X+18*randn(size(X));
figure;
image(x),colormap(map);
%使用 wdencmp()函数进行图像降噪
%寻找默认值
[thr,sorh,keepapp]=ddencmp('den','wv',x);
%使用全局阈值进行图像降噪
xd=wdencmp('gbl',x,'sym4',2,thr,sorh,keepapp);
figure;
image(xd),colormap(map);
```



结果如图 9-21 所示。可以看出，经过小波降噪后的图像非常清楚，达到了很好的平滑效果。


a)


b)

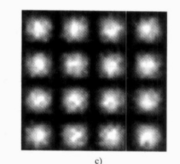

c)

图 9-21 小波的图像降噪应用

a) 原始图像 b) 噪声图像 c) 降噪后的图像

9.4 小波的融合技术

图像融合是将同一对象的两个或更多的图像合成为一幅图像，以便它比原来的任何一幅图像更容易被人们理解。这一技术可应用于多频谱图像理解及医学图像处理等领域，在这些场合，同一物体部件的图像往往是采用不同的成像机理得到的。

图像融合是信息融合的一个重要分支，广泛地用于目标识别、机器视觉、智能系统、医学图像处理等领域。传统图像融合方法主要是在时间域通过算术运算实现融合，具有算法简单直观，融合速度快，适合实时处理等优点，但其没有对频率变化进行考虑。

多分辨率图像融合算法则是在频域实现了图像的融合。根据分解形式的不同，多分辨率图像融合算法又可分为多分辨金字塔方法和小波变换方法。近年来，基于小波变换的图像融合越来越受到重视。由于人的视网膜图像是在不同频带上分别以不同算子进行融合，而基于小波分解的图像融合也是以同样方式进行的，所以，其可以获得与人的视觉特性更为接近的融合效果。可以认为，基于小波变换的图像融合应该是很有前途的研究方向。

以下简单介绍一下小波变换图像融合的基本原理。

如果一个图像进行 L 层小波分解，将得到（3L+1）层子带，其中包括低频的基带 C_j 和 $3L$ 层的高频子带 D^h 、D^v 和 D^d 。用 $f(x,y)$ 代表原始图像，记为 C_0 ，设尺度系数 $\Phi(x)$ 和小波系数 $\psi(x)$ 对应的滤波器系数矩阵分别为 $\boldsymbol{H}$ 和 $\boldsymbol{G}$，则二维小波分解算法可描述为

$$\begin{cases} C_{j+1} = \boldsymbol{H}C_j\boldsymbol{H}^{\mathrm{T}} \\ D_{j+1}^h = \boldsymbol{G}C_j\boldsymbol{H}^{\mathrm{T}} \\ D_{j+1}^v = \boldsymbol{H}C_j\boldsymbol{G}^{\mathrm{T}} \\ D_{j+1}^d = \boldsymbol{G}C_j\boldsymbol{G}^{\mathrm{T}} \end{cases} \tag{9-47}$$

式中，j 表示分解层数；h、v、d 分别表示水平、垂直、对角分量；$\boldsymbol{H}^{\mathrm{T}}$ 和 $\boldsymbol{G}^{\mathrm{T}}$ 分别是 $\boldsymbol{H}$ 和 $\boldsymbol{G}$ 的共轭转置矩阵。



小波重构算法为为

$$C_{j-1}=\boldsymbol{H}^{\mathrm{T}}C_j\boldsymbol{H}+\boldsymbol{G}^{\mathrm{T}}D_j^h\boldsymbol{H}+\boldsymbol{H}^{\mathrm{T}}D_j^v\boldsymbol{G}+\boldsymbol{G}^{\mathrm{T}}D_j^d\boldsymbol{G} \tag{9-48}$$

基于二维 DWT 的融合过程如图 9-22 所示，ImageA 和 ImageB 代表两幅原始图像 A 和 B，ImageF 代表融合后的图像，具体步骤如下：

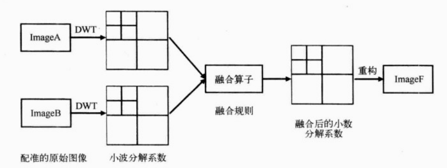


图 9-22　基于二维 DWT 的图像融合过程

1）图像的预处理。

图像滤波：对失真变质的图像直接进行融合，必然导致图像噪声融入融合效果，所以在进行融合前，必须对原始图像进行预处理以消除噪声。

图像配准：多种成像模式或多焦距提供的信息常常具有互补性，为了综合使用多种成像模式和多焦距以提供更全面的信息，常常需要将有效信息进行整合，使多幅图像在空间域中达到几何位置的完全对应。

2）对 ImageA 和 ImageB 进行二维 DWT 分解，得到图像的低频分量和高频分量。

3）根据低频分量和高频分量的特点，按照各自的融合算法进行融合。

4）对以上得到的高、低频分量，经过小波逆变换重构得到融合图像 ImageF。

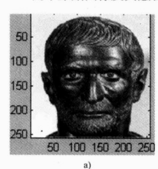
以下为用小波分析对两个不同的图像进行融合的应用举例程序代码。

```
load bust %载入原始图像 1
X1=X;
map1=map;
%画出 woman 图像
subplot(2,2,1);
image(X1);colormap(map);
axis square
%载入原始图像 2
load mask
X2=X;
map2=map;
for i=1:256
    for j=1:256
        if(X2(i,j)>100)
            X2(i,j)=1.2*X2(i,j);
        else
```

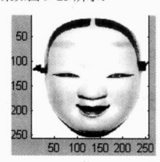


```
                X2(i,j)=0.5*X2(i,j);
            end
        end
    end
    %画出 wbarb 图像
    subplot(2,2,2);
    image(X2);colormap(map2);
    axis square
    [c1,s1]=wavedec2(X1,2,'sym4');
    sizec1=size(c1);
    for i=1:sizec1(2)
        c1(i)=1.2*c1(i);
    end
    [c2,s2]=wavedec2(X2,2,'sym4');
    c=c1+c2;
    c=0.5*c;
    s=s1+s2;
    s=0.5*s;
    xx=waverec2(c,s,'sym4');
    %画出融合后的图像
    subplot(2,2,3);image(xx);
    axis square
```

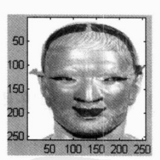
两个原始图像及其融合的结果如图 9-23 所示。



a)



b)



c)

图 9-23 小波的图像融合效果

a) 原始图像 1 b) 原始图像 2 c) 融合后的图像

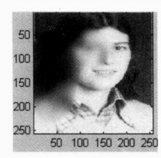
一幅图像和它某一部分放大后的图像融合，融合后的图像给人一种朦朦胧胧梦幻般的感觉，对较深的背景部分则做了淡化处理。

下面为利用图像融合方法从模糊图像中恢复图像的 MATLAB 程序代码。

```
    %载入第一幅模糊图像
    load cathe_1;
    X1=X;
    %载入第二幅模糊图像
```

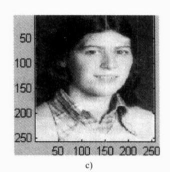
```
load cathe_2;
X2=X;
%基于小波分解的图像融合
XFUS=wfusimg(X1,X2,'sym4',5,'max','max');
%显示
colormap(map);
subplot(2,2,1);image(X1);
axis square;
subplot(2,2,2);image(X2);
axis square;
subplot(2,2,3);image(XFUS);
axis square;
```

执行程序代码，结果如图 9-24 所示。



a)

b)



c)

图 9-24 图像融合恢复图像

a) 原始图像 1 b) 原始图像 2 c) 融合图像

9.5 小波包在图像边缘检测中的应用

小波包分解后得到的图像序列由近似部分和细节部分组成，近似部分是原始图像对高频部分进行滤波所得的近似表示。经滤波后，近似部分去除了高频分量，因此能够检测到原始图像中所检测不到的边缘。

下面为利用小波包分解检测图像边缘的应用举例程序代码。

```
%载入并显示原始图像
load bust
%加入含噪
init=2055615866;
randn('seed',init);
X1=X+20*randn(size(X));
subplot(2,2,1);image(X1);
colormap(map);
axis square;
```



```
%利用小波 db4 对图像 X 进行一层小波包分解
T=wpdec2(X1,1,'db4');
%重构图像近似部分
A=wprcoef(T,[1 0]);
subplot(2,2,2);image(A);
axis square;
%检测边缘
%原始图像的边缘检测
B1=edge(A,'sobel');
subplot(2,2,3);imshow(B1);
axis square;
%图像近似部分的边缘检测
B2=edge(X1,'sobel');
subplot(2,2,4),imshow(B2);
axis square
```

执行程序代码，效果如图 9-25 所示。

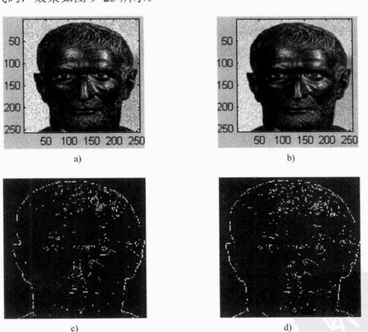


图 9-25　图像边缘检测结果

a) 原始的含噪图像　b) 图像近似部分　c) 近似部分边缘　d) 原始图像的边缘

对近似图像进行边缘检测的结果和直接对原始图像进行边缘检测的结果相比，前一种方法的效果更好。

需要指出的是，这里采用的是一种最为简单的边缘检测算法。实际上，基于小波的图像边缘检测技术发展非常迅速，近些年已经诞生了很多种新颖的技术和方法，如多尺度小波变换边界提取算法、嵌入可信度的边缘检测方法、奇异点模极大值检测算法等。感兴趣的读者

可以查阅其他相关资料。

9.6 小波包与图像消噪

在本节中，小波包分析进行图像消噪处理的基本原理和方法与前面分绍的对信号消噪的相同。本节仅以具体的实例来说明小波包分析在图像消噪处理中的应用。

在 MATLAB 的小波工具箱中，提供了一个 wpdencmp()函数，它是专门利用小波包分解实现消噪和压缩处理的。

Wpdencmp()函数的语法格式为

- [XD,TREED,PERFO,PERFL2]=wpdencmp(X,SORH,N,'wname',CRIT,PAR,KEEPAPP)
- [XD,TREED,PERFO,PERFL2]=wpdencmp(TREE,SORH,CRIT,PAR,KEEPAPP)

【使用说明】利用小波包实现信号和图像的消噪或压缩。

输入参数：SORH 指定选取软阈值（SORH='S'）或硬阈值（SORH='H'）；N 为小波分解的层数；wname 指定分解时所用的小波；CRIT 和 PAR 定义了熵准则；TREE 是小波包分解树结构。

输出参数：PERFO,PERFL2 返回压缩比例系数。

利用小波包变换对一个二维含噪图像进行消噪处理。

```
%装载并显示原始图像
load F:\image\fs13
subplot(2,2,1);
image(X);
colormap(map);
title('原始图像');
axis square;

%在图像中加入噪声
init=2055615866;
randn('seed',init);
X1=X+10*randn(size(X));
subplot(2,2,2);
image(X1);
colormap(map);
title('含噪图像');
axis square;

%基于小波包的消噪处理
thr=10;sorh='s';
crit='shannon';
keepapp=0;
X2=wpdencmp(X1,sorh,3,'sym4',crit,thr,keepapp);

%画出消噪后的图像
subplot(2,2,3);
```



```
image(X2);
colormap(map);
title('全局阈值消噪图像');
axis square;

%对图像进行平滑处理以增强消噪效果（中值滤波）
for i=2:175;
    for j=2:259
        Xtemp=0;
        for m=1:3
            for n=1:3
                Xtemp=Xtemp+X2(i+m-2,j+n-2);
            end
        end
        Xtemp=Xtemp/9;
        X3(i,j)=Xtemp;
    end
end

%显示平滑结果
subplot(2,2,4);
image(X3);
colormap(map);
title('平滑后的图像');
axis square;
```

计算结果如图 9-26 所示，其中图 9-26a 是原始的图像，图 9-26b 是添加噪声后的图像，通过小波包分解并设置全局阈值，消噪后的结果如图 9-26c 所示，与含噪图像相比，明显清楚了很多；进一步对消噪后的图像进行平滑处理，如图 9-26d 所示，与消噪后的图像相比，它明显光滑了。

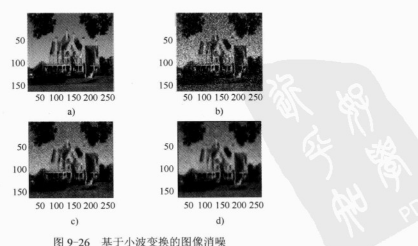


图 9-26 基于小波变换的图像消噪

a) 原始图像 b) 含噪图像 c) 全局阈值消噪图像 d) 平滑后的图像

除了利用函数 wpdencmp()对图像进行消噪外，还可以利用二维小波包分解函数 wpdec2()来实现图像消噪。

wpdec2()函数的语法格式为

- T=wpdec2(X,N,'wname',E,P)
- T=wpdec2(X,N,'wname')

【使用说明】二维小波包分解。

输入参数：X 是分析矩阵；N 是分解的层数；wname 是小波基函数；E 是熵的类型；P 是依赖于 E 的可选参数。

输出参数：T 返回小波包树。

利用二维小波包分解对一个二维含噪图像进行消噪处理。

```
%装载并显示原始图像
load F:\image\fs06
subplot(2,2,1);
image(X);
colormap(map);
title('原始图像');
axis square;

%生成含噪图像
init=2055615866;
randn('seed',init);
X1=X+20*randn(size(X));
subplot(2,2,2);
image(X1);
colormap(map);
title('含噪图像');
axis square;

%用小波 sym2 对图像 X1 进行一层小波包分解
T=wpdec2(X1,1,'sym2');

%设置阈值
thr=8.342;

%对图像的小波包分解系数进行软阈值量化
NT=wpthcoef(T,0,'s',thr);

%仅对低频系数进行重构
X2=wprcoef(NT,1);

%画出消噪后的图像
subplot(2,2,3);
image(X2);
colormap(map);
```



```
title('消噪后的图像');
axis square;
```

计算结果如图 9-27 所示，其中图 9-27a 是原始的图像，图 9-27b 是添加噪声后的图像，通过二维小波包分解后并对系数进行阈值化处理后，重构的图像如图 9-27c 所示，与含噪图像相比，它明显清楚了很多，达到了消噪的效果。

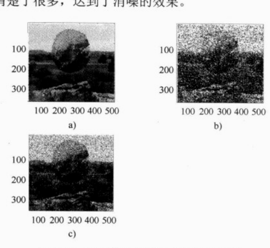


图 9-27　基于二维小波包分解的图像消噪

a) 原始图像　b) 含噪图像　c) 消噪后的图像

9.7　小结

本章主要讲述了小波技术在图像处理中的应用。首先介绍了小波分析的基础知识，包括一维小波变换的基本原理、二维小波变换和多分辨率分析以及小波工具箱的图像处理功能。然后介绍了小波用于图像降噪、图像压缩、图像增强和图像融合的基本原理和 MATLAB 实现方法。本章介绍的大多是最基本和简单的用法。由于小波分析和图像处理技术都在不断发展之中，所以新的研究内容和处理算法层出不穷，读者可以针对感兴趣的内容查询相关的资料以供进一步研究。

第 10 章　图像特征的描述

图像描述是图像处理的核心内容。为了便于有效地研究和应用，往往需要用一些简单明确的数值、符号或图来表征给定的图像及已分割的图像区域。这些数值、符号或图是按一定的概念和公式产生的，反映了图像或图像区域的基本信息和主要特征。通常称这些数值、符号或图为图像的特征，而用这些特征表示图像称为图像描述。

10.1　灰度描述

10.1.1　幅度特征

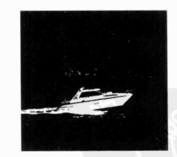
在所有的图像特征中，最基本的是图像的幅度特征。可以在某一像素点或其领域内做出幅度的测量，如在 $N\times N$ 区域内的平均幅度，即

$$\overline{f}(x,y)=\frac{1}{N^2}\sum_{i=0}^{N}\sum_{j=0}^{N}f(i,j) \tag{10-1}$$

可以直接从图像像素的灰度值，或从某些线性、非线性变换后构成新的图像幅度的空间来求得各式各样图像的幅度特征图。图像的幅度特征对于分离目标物的描述等具有十分重要的作用。如图 10-1 所示，其中图 10-1a 是原始图像，图 10-1b 是利用幅度特征将背景中的快艇分割出来的结果。



a)



b)

图 10-1　灰度信息对目标进行分割

a) 原始图像　b) 幅度特征分割结果

10.1.2　直方图特征

一幅数字图像可以看成是一个二维随机过程的一个样本，可以用联合概率分布来描述。通过对图像的各像素幅度值可以设法估计出图像的概率分布，从而形成图像的直方图特征。

图像灰度的一阶概率分布定义为

$$P(b)=P\{f(x,y)=b\} \quad (0\leqslant b\leqslant L-1) \tag{10-2}$$

式中，b 为量化值；L 为量化值范围，即

$$P(b)\approx\frac{N(b)}{M} \tag{10-3}$$

式中，M 为围绕 (x,y) 点被测窗口内的像素总数；$N(b)$ 为该窗口内灰度值为 b 的像素总数。

图像的直方图特征可以提供图像信息的许多特征。例如，若直方图密集地分布在很窄的区域内，说明图像的对比度很低；若直方图有两个峰值，则说明存在着两种不同亮度的区域。

一阶直方图的特征参数有

平均值：$\bar{b}=\sum_{b=0}^{L-1}bP(b)$

方差：$\sigma_b^2=\sum_{b=0}^{L-1}(b-\bar{b})^2P(b)$

倾斜度：$b_n=\frac{1}{\sigma_b^3}\sum_{b=0}^{L-1}(b-\bar{b})^3P(b)$

峭度：$b_k=\frac{1}{\sigma_b^4}\sum_{b=0}^{L-1}(b-\bar{b})^4P(b)-3$

能量：$b_N=\sum_{b=0}^{L-1}[P(b)]^2$

熵：$b_E=-\sum_{b=0}^{L-1}P(b)\log_2[P(b)]$

二阶直方图特征是以像素对的联合概率分布为基础得出的。若两个像素 $f(i,j)$ 及 $f(m,n)$ 分别位于 (i,j) 点和 (m,n) 点，两者的间距为 $|i-m|$、$|j-n|$，并可用极坐标 ρ、θ 表达，那么其幅度值的联合分布为

$$P(a,b)\underline{\underline{\Delta}}P_k\{f(i,j)-a,f(m,n)-b\} \tag{10-4}$$

式中，a、b 为量化的幅度值。因此，直方图估值的二阶分布为

$$P(a,b)\approx\frac{N(a,b)}{M} \tag{10-5}$$

式中，$N(a,b)$ 表示在图像中，在 θ 方向上、径向间距为 ρ 的像素对 $f(i,j)=a$，$f(m,n)=b$ 出现的频数；M 为测量窗口中像素的总数。

假设图像的各像素对都是相互关联的，则 $P(a,b)$ 将在陈列的对角线上密集起来。以下列出的一些度量，用来描述围绕 $P(a,b)$ 对角线能量扩散的情况。

自相关：$B_A=\sum_{a=0}^{L-1}\sum_{b=0}^{L-1}abP(a,b)$

协方差：$B_C=\sum_{a=0}^{L-1}\sum_{b=0}^{L-1}(a-\bar{a})(b-\bar{b})P(a,b)$

惯性矩：$B_I = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1}(a-b^2)P(a,b)$

绝对值：$B_V = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1}|a-b|P(a,b)$

能量：$B_N = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1}[P(a,b)]^2$

熵：$B_E = -\sum_{a=0}^{L-1}\sum_{b=0}^{L-1}P(a,b)\log_2[p(a,b)]$

10.1.3 变换系数的特征

由于图像的二维变换得出的系数反映了二维变换后图像在频域的分布情况，因此常常用二维的傅里叶变换作为一种图像特征的提取方法。例如：

$$F(u,v) = \iint f(x,y)\mathrm{e}^{-\mathrm{j}2\pi(ux+vy)}\mathrm{d}x\mathrm{d}y \tag{10-6}$$

设 $M(u,v)$ 是 $F(u,v)$ 的平方值，即

$$M(u,v) = |F(u,v)|^2 \tag{10-7}$$

当 $f(x,y)$ 的原点有了位移时，$M(u,v)$ 的值保持不变，因此 $M(u,v)$ 与 $F(u,v)$ 不是唯一对应的，这种性质称为位移不变性，在某些应用中可利用这一特点。

如果把 $M(u,v)$ 在某些规定区域内的累计值求出，也可以把图像的某些特征突出起来，这些规定的区域如图 10-2 所示，其中图 10-2a 为水平切口，图 10-2b 为垂直切口，图 10-2c 为环形切口。

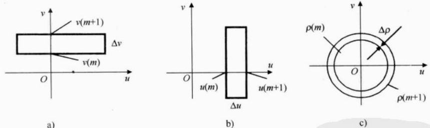


图 10-2 不同类型的切口

a) 水平切口 b) 垂直切口 c) 环形切口

由各种不同切口规定的特征度量可由下面各式来定义。

水平切口：$S_1(m) = \int_{v(m)}^{v(m+1)} M(u,v)\mathrm{d}v$

垂直切口：$S_2(m) = \int_{u(m)}^{u(m+1)} M(u,v)\mathrm{d}u$

环状切口：$S_3(m) = \int_{\rho(m)}^{\rho(m+1)} M(\rho,\theta)\mathrm{d}\rho$

式中，$M(\rho,\theta)$ 为 $M(u,v)$ 的极坐标式。

这些特征说明了图像中含有这些切口的频谱成分的含量。把这些特征提取出来以后，可以作为模式识别或分类系统的输入信息。这种方法已经成功地运用到土地情况分类、放射照片病情诊断等方面。

10.2 纹理分析

目前还没有统一和公认的有关纹理的确切定义。一般情况下把类似于布纹、草地、砖头、墙面等具有重复性结构的图像称为纹理图像。纹理图像在局部区域内可能呈现不规则性，但整体上则表现出一定的规律性，其灰度分布往往表现出某种周期性。纹理图像所表现出的这种特有的性质称为纹理。实际中很多图像具有纹理型结构，对这类纹理型图像可以通过纹理分析取其宏观特征信息。

10.2.1 纹理特征

纹理一词最初指纤维物的外观，纹理图像在很大范围内没有重大细节变化，在这些区域内图像往往显示出重复性结构。有时，物体在纹理上与其周围背景和其他物体有区别、这时图像分割应以纹理为基础。虽然纹理目前尚无统一的定义，但一般来说，纹理是由许多相互接近的、互相交织的元素构成，它们具有周期性。纹理在一定程度上反映了一个区域中像素灰度级的空间分布的属性。

纹理可分为人工纹理和天然纹理（自然纹理）。人工纹理由某种符号的有序排列组成，这些符号可以是线条、点、字母、数字等。自然纹理是具有重复排列现象的自然景象，如砖墙、种子、森林、草地之类的照片。图 10-3a 是常见的人工纹理结构，图 10-3b 是常见的自然纹理结构。由此可见，人工纹理往往是有规则的，而自然纹理往往是无规则的。

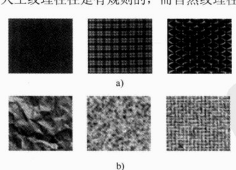


图 10-3 典型的纹理结构图

a) 人工纹理 b) 自然纹理

归纳起来，对纹理的认识有两种看法。一种是凭人们的直观印象；而另一种是凭图像本身的结构。从直观印象出发的方法包含有心理学因素，这样就会产生多种不同的统计纹理特性，从这一观点出发，纹理分析应该采用统计法；从图像结构观点出发，则认为纹理是一种

结构，根据这一观点，纹理分析应该采用句法结构法。

描述图像特性的参数有很多种，对纹理图像来说有必要知道各个像素及其邻近像素的灰度分布情况。了解邻近像素灰度值变化情况的最简单方法是取 1 阶微分、2 阶微分的平均值与方差，如果要考虑纹理的方向性特征，则可考察 θ 方向与 $(\theta+\pi/2)$ 方向差分的平均值与方差。纹理分析常用的方法有统计法、自相关函数法、频谱法、句法结构法和联合概率矩阵法等。

10.2.2 统计法

统计法是利用图像内某一区域或物体的灰度直方图的纹理结构进行描述，它又可以分为灰度差分统计法和行程长度统计法。

1. 灰度差分统计法

取图像内任意一点 (x,y)，设与该点相邻的点 $(x+\Delta x, y+\Delta y)$ 的灰度差值为

$$g_{\Delta}(x,y)=g(x,y)-g(x+\Delta x, y+\Delta y) \tag{10-8}$$

式中，$g_{\Delta}(x,y)$ 称为灰度差分。若 $g_{\Delta}(x,y)$ 的所有可能取值共有 m 级，则令 (x,y) 在整个区域上移动，统计出 $g_{\Delta}(x,y)$ 取各个灰度级的次数，由此作出 $g_{\Delta}(x,y)$ 的直方图。根据直方图可以得出 $g_{\Delta}(x,y)$ 取不同灰度值的概率 $p_{\Delta}(i)$。

若 i 取值较小，而概率 $p_{\Delta}(i)$ 较大，则说明纹理较粗糙，若概率 $p_{\Delta}(i)$ 较平坦，则说明纹理较细密。

灰度差分统计法一般采用以下参数描述纹理图像的特征。

对比度：
$$CON=\sum_i i^2 p_{\Delta}(i) \tag{10-9}$$

角度方向二阶矩：
$$ASM=\sum_i [p_{\Delta}(i)]^2 \tag{10-10}$$

熵：
$$ENT=-\sum_i p_{\Delta}(i)\lg p_{\Delta}(i) \tag{10-11}$$

平均值：
$$MEAN=\frac{\sum_i i p_{\Delta}(i)}{m} \tag{10-12}$$

根据上述公式，若 $p_{\Delta}(i)$ 较平坦，则 ASM 较小，而 ENT 较大；若 $p_{\Delta}(i)$ 分布在原点附近，则 $MEAN$ 较小。

2. 行程长度统计法

设图像内任意一点 (x,y) 的灰度值为 g，与其相邻点的灰度值也可能为 g 或其他值，统计出从任意一点出发沿 θ 方向上连续 n 个点都具有灰度值 g 所发生的概率，记此概率为 $P(g,n)$。在同一方向上具有相同灰度值的像素点的数量称为行程长度。根据 $P(g,n)$ 可以定义以下参数来描述纹理特征。

行程加重法：
$$LRE=\frac{\sum_{g,n} n^2 p(g,n)}{\sum_{g,n} p(g,n)} \tag{10-13}$$

灰度值分布：
$$LRE=\frac{\sum_{g}\left[\sum_{n}p(g,n)\right]^{2}}{\sum_{g,n}p(g,n)} \tag{10-14}$$

行程长度分布：
$$LRE=\frac{\sum_{g}\left[\sum_{n}p(g,n)\right]}{\sum_{g,n}p(g,n)} \tag{10-15}$$

行程行：
$$LRE=\frac{\sum_{g,n}p(g,n)}{N^{2}} \tag{10-16}$$

式中，N^2 为图像的像素总数。

10.2.3 自相关函数法

物体的纹理常用其粗糙性加以描述。例如，在相同的外观条件下，毛织品一般比丝织品粗糙。粗糙性的程度与局部结构的空间重复周期性有关，周期大的纹理细；反之则纹理粗糙。这种感觉上的粗糙虽然不足以定量表示纹理的测度，但可说明纹理测度的变化趋势，即纹理测度小表示纹理比较细密；纹理测度值大表示纹理比较粗糙。

设图像以 $f(m,n)$ 表示，则自相关函数可定义如下：

$$C(\varepsilon,\eta,j,k)=\frac{\sum_{m=j-w}^{j+w}\sum_{n=k-w}^{j+w}f(m,n)f(m-\varepsilon,n-\eta)}{\sum_{m=j-w}^{j+w}\sum_{m=j-w}^{k+w}[f(m,n)]^{2}} \tag{10-17}$$

式（10-17）是对 $(2w+1)\times(2w+1)$ 窗口内的每一个像素点 (j,k) 与偏离值为 $\varepsilon,\eta=0,\pm1,\pm2,\cdots,\pm T$ 的像素之间的相关值进行计算。一般纹理区对给定偏离 (ε,η) 量的相关性要比细纹理区高，因而纹理粗糙性与自相关函数的扩展成正比。自相关函数扩展的一种测度就是二阶矩，定义如下：

$$T(j,k)=\sum_{\varepsilon=-T}^{j}\sum_{\eta=-T}^{j}\varepsilon^{2}\ \eta^{2}C(\varepsilon,\eta,j,k) \tag{10-18}$$

纹理越粗糙，则 $T(j,k)$ 越大。因此，可以用 $T(j,k)$ 作为度量纹理粗糙程度的参数之一。

10.2.4 频谱法

频谱法即傅里叶频谱法，指依据傅里叶频谱的频率特性来描述周期的或近似周期的二维图像纹理结构。对于图像而言，二维傅里叶变换如下：

$$F(u,v)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(x,y)\mathrm{e}^{-\mathrm{j}2\pi(ux+vy)}\mathrm{d}x\mathrm{d}y \tag{10-19}$$

二维傅里叶变换的功率谱如下：

$$E=|F(u,v)|=FF^{*} \tag{10-20}$$

傅里叶频谱中突起的峰值对应纹理模式的主方向，峰值在频域平面的位置对应模式的基本周期，若采用滤波将周期性成分滤除，则剩下的非周期性部分可用统计法描述。

实际应用中，一般将频谱先转换到极坐标系中，如图 10-4 所示，此时傅里叶变换可用 $F(r,\theta)$ 表示，其频谱可用函数 $S(r,\theta)=|F(r,\theta)|^2$ 表示。

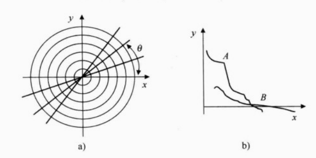


图 10-4　纹理和对应的频谱示意图

对每个确定的方向 θ，$S(r,\theta)$ 是一个一维函数 $S(r)$；对每个确定的频率 r，$S(r,\theta)$ 是一个一维函数 $S(\theta)$。对给定的 θ，分析 $S(r)$ 可得到频谱沿原点射出方向的行为特性；对给定的 r，分析 $S(\theta)$ 可得到频谱在以原点为中心的圆上的行为特性。如果将这些函数积分，则可得到更为全局性的描述，即

$$\begin{cases} S(r)=\int_0^{2\pi}[F(r,\theta)]^2\mathrm{d}\theta \\ S(\theta)=\int_0^{2\pi}[F(r,\theta)]^2\mathrm{d}r \end{cases} \tag{10-21}$$

如果是离散图像，则将上述积分分别以求和代替，即可得出相应的公式，即将这些函数序列分别对 θ 或 r 求和，同样可得到沿某一角度 θ 方向和半径 r 方向的全局性的描述，即

$$S(\theta)=\sum_{\theta}|F(r,\theta)|^2=\sum_{\theta}S(r,\theta) \tag{10-22}$$

$$S(r)=\sum_{r}|F(r,\theta)|^2=\sum_{r}S(r,\theta) \tag{10-23}$$

$S(r)$ 和 $S(\theta)$ 构成整个图像或图像区域的纹理频谱能量描述。如果 r 较小，$S(r)$ 很大；或 r 很大，$S(r)$ 却较小，则说明是粗糙性纹理结构；反之，如果 r 变化对 $S(r)$ 的影响较小，则为细密的纹理结构。在纹理粗糙的情况下，能量多集中在离原点很近的范围内，即图 10-4b 中的曲线 A，而对于纹理较细的情况下，能量分布在离原点较远的范围内，即图 10-4b 中的曲线 B。

10.2.5 纹理的句法结构分析法

在纹理的句法结构分析中，将纹理定义为结构基元按某种规则重复分布所构成的模式。进行纹理结构分析，需要先描述结构基元的分布规律。因此，一般可进行如下两项工作。

1）从输入图像中提取结构基元并描述其特征。

2）描述结构基元的分布规则。具体方法如下：

首先将纹理图像分成许多窗口，即形成子纹理。其中最小的小块就是最基本的子纹理（即基元）。纹理基元可以是一个像素，也可以是 4 个或 9 个灰度比较一致的像素集合。纹理的表达可以是多层次的，如图 10-5a 所示，它可以从像素或小块纹理一层一层地向上拼合。而且，基元的排列可有不同规则，如图 10-5b 所示，第一级纹理排列为 ABA，第二级排列为 BAB 等，其中 A、B 代表基元或子纹理，于是就组成了一个多层的树状结构，可用树状文法产生一定的纹理，并以句法加以描述。

纹理的树状排列可有多种方法。如图 10-5d 所示，树根安排在中间，树枝向两边伸出，每个树枝有一定的长度。又如图 10-5c 所示，树根安排在一侧，分枝都向另一侧伸展。

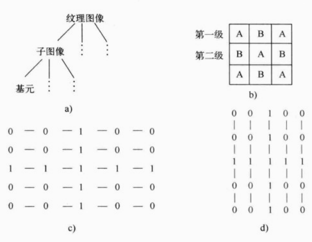


图 10-5 纹理的树状描述及排列

纹理判别可用以下方法：

首先将纹理图像分成固定尺寸的窗口，用树状文法说明属于同纹理图像的窗口，可以用树状自动机识别树状，因此，对每一个纹理文法可建立一个“结构保存的误差修正树状自动机”。该自动机不仅可以接受每个纹理图像中的树，而且能用最小距离判据辨识类似的有噪声的树。以后，可以对任意一个分割成窗口的输入图像进行分类。

10.2.6 联合概率矩阵法

联合概率矩阵法是通过对图像的所有像素进行统计并描述其灰度分布的一种方法。取图像中任意一点 (x,y) 及偏离它的另一点 $(x+a,y+b)$ 组成一个点对，设该点对的灰度值为 (g_1,g_2)。令点 (x,y) 在所分析的区域内移动，则可得到全部的 (g_1,g_2) 值。若灰度值的级数为 k，则 (g_1,g_2) 共有 g^2 种组合。对于整个区域，统计出每一个 (g_1,g_2) 值出现的次数，然后排成一个方阵，再用 (g_1,g_2) 出现的总数将其归一化为出现的概率 $p(g_1,g_2)$，这样的方阵称为联合概率矩阵，又称为灰度共生矩阵或灰度共现矩阵。

图 10-6 为一个简单的例子。图 10-6a 为原始图像，共有 16 个灰度级，为使概率矩阵简单起见，首先将图 10-6a 的灰度级减为 4 级，变为图 10-6b 的形式。(g_1,g_2) 分别取值为 0、1、2、3，故可以将 (g_1,g_2) 各种组合出现的次数排列起来，可以得到如图 10-6c～e 所示的联合概率矩阵，以图 10-6c 中的频数 10 为例，该数据表示图 10-6b 中灰度值为（0,1），



共出现 10 次，其余可以类推。

由此可见，差分值（a,b）取不同的数值，就可以得到不同情况下的联合概率矩阵。（a,b）取值要根据纹理周期分布的特性进行选择，对于较细的纹理，选取（0,1）、（1,1）、（2,0）等较小的差分值，即 a、b 取值较小时，对应于变化缓慢的纹理结构，其联合概率矩阵对角线上的数值较大。若纹理的变化越快，则对角线上的数值就越小，对角线两侧的元素值却增大。为了能描述纹理的状况，有必要选取能综合反映联合概率矩阵状况的参数，典型的有如下几种：

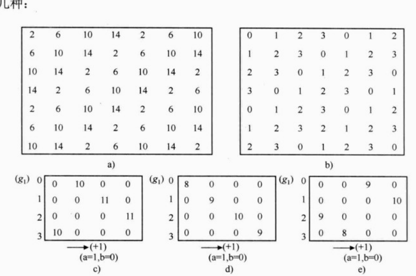


图 10-6　灰度共生矩阵计算方法

（1）角二阶矩

角二阶矩是图像灰度均匀性的一种度量。

$$Q_1 = \sum_{g_1}\sum_{g_2}[p(g_1,g_2)]^2 \tag{10-24}$$

（2）惯性矩

惯性矩又称为对比度，可以理解为图像的清晰度，在图像中纹理的纹沟越深，对比度则越大，图像的视觉效果越清晰。

$$Q_2 = \sum_k k^2\left[\sum_{g_1}\sum_{g_2} p(g_1,g_2)\right] \tag{10-25}$$

式中，$k = g_1 - g_2$。　　（10-26）

（3）相关性

相关性用于衡量灰度共生矩阵的元素在行或列方向上的相似程序。

$$Q_3 = \frac{\sum_{g_1}\sum_{g_2} g_1 g_2 p(g_1,g_2) - \mu_x\mu_y}{\sigma_x\sigma_y} \tag{10-27}$$

式中，
$$\begin{cases}\mu_x = \sum_{g_1} g_1 \sum_{g_2} p(g_1, g_2) \\ \mu_y = \sum_{g_2} g_2 \sum_{g_1} p(g_1, g_2)\end{cases} \quad \begin{cases}\sigma_x^2 = \sum_{g_1} (g_1 - \mu_x)^2 \sum_{g_2} p(g_1, g_2) \\ \sigma_y^2 = \sum_{g_2} (g_2 - \mu_y)^2 \sum_{g_1} p(g_1, g_2)\end{cases} \tag{10-28}$$

（4）熵

熵是图像具有的信息量的一种度量，纹理信息也是图像的信息，若一幅数字图像没有纹理，则灰度共生矩阵接近为零矩阵。

$$Q_4 = -\sum_{g_1}\sum_{g_2} p(g_1, g_2) \lg p(g_1, g_2) \tag{10-29}$$

尽管 Q_1、Q_2、Q_3、Q_4 代表的图像特征并不是很直观，但它们是描述纹理特征非常有效的参数。

10.3 形状描述

10.3.1 链码

1. 链码简介

链码（Chain Code）在图像处理和模式识别中是常用的一种表示方法，它最初是由 Freemam 于 1961 年提出来的，用来表示线条模式，至今它仍被广泛使用。根据链的斜率不同，常用的有 4 方向和 8 方向链码，其方向定义分别如图 10-7a 和图 10-7b 所示。在 4 方向链码中，4 个方向码的长度都是一个像素单位；在 8 方向链码中，水平和垂直方向的方向码的长度都是一个像素单位，而对角线方向的 4 个方向码为像素单位的 $\sqrt{2}$ 倍。因此，它们的共同特点是直线段的长度固定，方向数有限，故可以利用一系列具有这些特点的相连的直线段来表示目标的边界，这样只有边界的起点需要用绝对坐标表示，其余点都可只用接续方向来代表偏移量。由于表示一个方向数比表示一个坐标值所需比特数少，而且对每一个点又只需要一个方向数就可以代替两个坐标值，因此，链码表达可大大减少边界表示所需的数据量，所以常用链码来作为对边界点的一种编码表示方法。

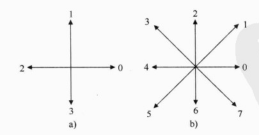


图 10-7 链码值与方向的对应关系

从在物体边界上任意选取的某个起始点坐标开始，跟踪边界并赋给每两个相邻像素的连线一个方向值，最后按照逆时针方向沿着边界将这些方向码连接起来，就可以得到链码。因此，链码的起始位置和链码完整地包含了目标的形状和位置信息。

例如，在如图 10-8 所示的以 a 为起点、以箭头为走向的闭合边界（小圆点处表示各像素点）中，其 8 方向链码为 001711222433445676656。

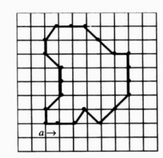


图 10-8　以 a 为起点，以箭头为走向的闭合边界

使用链码时，起点的选择是很关键的。对同一个边界，如用不同的边界点作为链码的起点，得到的链码是不同的。为解决这个问题可把链码归一化，具体做法为：给定一个从任意点开始产生的链码，把它看做一个由各方向数构成的自然数。首先，将这些方向数按一个方向循环，以使它们所构成的自然数的值最小；然后，将这样转换后所对应的链码起点作为这个边界的归一化链码的起点。

2. 链码的旋转不变性

用链码表示给定目标的边界时，如果目标平移，链码不会发生变化；而如果目标旋转，则链码会发生变化。为解决这个问题，可利用链码的一阶差分来重新构造一个表示原链码各段之间方向变化的新序列，这相当于把链码进行旋转归一化。差分可用相邻的两个方向数按反方向相减（后一个减去前一个）得到，如图 10-9 所示。上面一行为原链码（括号中为最右面的方向数循环到左边），下面一行为上面一行的数两两相减得到的差分码。左边的目标在逆时针旋转 90° 后成为右边的形状，可见，原链码发生了变化，但差分码并没有变化。

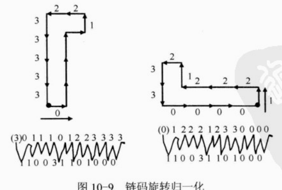


图 10-9　链码旋转归一化

10.3.2 傅里叶描述子

对边界的离散傅里叶变换表达可以作为定量描述边界形状的基础。采用傅里叶描述的一

个优点是将二维问题简化为一维问题，即将 xy 平面中的曲线段转化为一维函数 $f(r)$（在 $r-f(r)$ 平面上），也可将 xy 平面中的曲线段转化为复平面上的一个序列。具体就是将 xy 平面与复平面 uv 重合，其中，实部 u 轴与 x 轴重合，虚部 v 轴与 y 重合，这样可用复数 u+jv 的形式来表示给定边界上的每个点（x,y）。这两种表示在本质上是一致的，是点与点对应的，如图 10-10 所示。

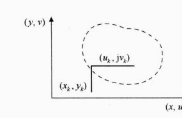


图 10-10　边界点的两种表示方法

对于 xy 平面上一个由 K 个点组成的边界来说，任意选取一个起始点 (x_0,y_0)。然后沿着顺时针方向绕行一周，可以得到一个点序列：$(x_0,y_0),(x_1,y_1),\cdots,(x_{K-1},y_{K-1})$。如果记 $x(k)=x_k, y(k)=y_k$，并把它们用复数形式表示，则得到一个坐标序列：

$$s(k)=x(k)+\mathrm{j}y(k) \qquad (k=0,1,\cdots,K-1) \tag{10-30}$$

$s(k)$ 的离散傅里叶变换为

$$S(u)=\sum_{k=0}^{K-1}s(k)\mathrm{e}^{-\mathrm{j}2\pi uk/K} \qquad (u=0,1,\cdots,K-1) \tag{10-31}$$

式中，傅里叶系数 $S(u)$ 可称为边界的傅里叶描述子，它的傅里叶逆变换为

$$s(k)=\frac{1}{k}\sum_{u=0}^{K-1}S(u)\mathrm{e}^{\mathrm{j}2\pi uk/K} \qquad (k=0,1,\cdots,K-1) \tag{10-32}$$

由于傅里叶变换的高频分量对应一些细节，而低频分量对应基本形状，因此只利用 $S(u)$ 的前 M 个系数来重构原来的图像，从而可以得到对 $s(k)$ 的一个近似而不改变其基本形状，即

$$\hat{s}(k)=\frac{1}{M}\sum_{u=0}^{M-1}S(u)\mathrm{e}^{\mathrm{j}2\pi uk/K} \qquad (k=0,1,\cdots,K-1) \tag{10-33}$$

10.3.3 形状特征的描述

1. 长轴和短轴

当物体的边界已知时，用其外接矩形的尺寸来刻画它的基本形状是最简单的方法，如图 10-11a 所示，求物体在坐标轴方向上的外接矩形，只需计算物体边界点的最大坐标值和最小坐标值，就可得到物体的水平跨度和垂直跨度。但是，对任意朝向的物体，水平和垂直不一定是我们感兴趣的方向，这时，就有必要确定物体的主轴，然后计算反映物体形状特征的主轴方向上的长度和与其垂直方向上的宽度，这样的外接矩形是物体的最小外接矩形（Minimum External Rectangle，MER），如图 10-11b 所示。

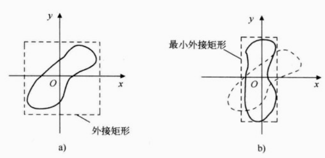


图 10-11 MER 法求物体的长轴和短轴

a) 坐标系方向上的外接矩形 b) 旋转物体使外接矩形最小

2. 矩形度

图像区域面积 A_0 与其最小外接矩形的面积 A_{MER} 之比即为矩形度：

$$R=\frac{A_0}{A_{MER}} \tag{10-34}$$

矩形度反映区域对其最小外接矩形的充满程度，当区域为矩形时，矩形度 $R=1.0$；当区域为圆形时，$R=\pi/4$；对于边界弯曲、呈不规则分布的区域，$0<R<1$。

3. 长宽比

长宽比 r 是将细长目标与近似矩形或圆形目标进行区分时，采用的形状度量。长宽比 r 为最小外接矩形的宽与长的比值，定义式如下：

$$r=\frac{W_{MER}}{L_{MER}} \tag{10-35}$$

4. 圆形度

圆形度用来刻画物体边界的复杂程度，有 4 种圆形度测度。

（1）致密度 C

致密度又称为复杂度，也称为分散度，其定义为区域周长（P）的平方与面积（A）的比：

$$C=\frac{P^2}{A} \tag{10-36}$$

致密度描述了区域单位面积的周长大小。致密度大，表明单位面积的周长大，即区域离散，为复杂形状；反之，致密度小，为简单形状。当图像区域为圆时，C 有最小值 4π；图像为其他任何形状的图像区域时，$C>4\pi$，且形状越复杂，C 值越大。例如，不管面积多大，正方形区域致密度 $C=16$，正三角形区域致密度 $C=12\sqrt{3}$。

（2）边界能量 E

假定物体的周长为 P，用变量 p 表示边界上的点到某一起始点的距离。边界上任意一点都有一个瞬时曲率半径 $r(p)$，它是该点与边界相切圆的半径，如图 10-12 所示。p 点的曲率函数为

$$K(p)=\frac{1}{r(p)} \tag{10-37}$$

函数 $K(p)$ 是周期为 P 的周期函数。

定义单位边界长度的平均能量：

$$E=\frac{1}{P}\int_0^p\left|K(p)^2\right|\,\mathrm{d}p \tag{10-38}$$

在面积相同的条件下，圆具有最小边界能量 $E_0=(2\pi/P)^2=(1/R)^2$，其中 R 为圆的半径。边界能量更符合人感觉上对边界复杂性的理解。

（3）圆形性

圆形性（Circularity）C 是一个用区域 R 的所有边界点定义的特征量，即

$$C=\frac{\mu_R}{\delta_R} \tag{10-39}$$

式中，μ_R 为区域重心到边界点的平均距离；δ_R 为区域重心到边界点的距离方均差。两者的值分别为

$$\mu_R=\frac{1}{K}\sum_{k=0}^{K-1}\left|(x_k,y_k)-(\overline{x},\overline{y})\right| \tag{10-40}$$

$$\delta_R=\frac{1}{K}\sum_{k=0}^{K-1}\left[\left|(x_k,y_k)-(\overline{x},\overline{y})\right|-\mu_R\right]^2 \tag{10-41}$$

当区域 R 趋向圆形时，特征量 C 是单调递增且趋向无穷的，它不受区域平移、旋转和尺度变化的影响，可以推广用于描述三维目标。

（4）面积与平均距离平方的比值

图形度的第 4 个指标利用了边界上的点到物体内部某点的平均距离 $\overline{d}$，即

$$\overline{d}=\frac{1}{N}\sum_{i=1}^{N}x_i \tag{10-42}$$

式中，x_i 为具有 N 个点的物体中的第 i 个点到与其最近的边界点的距离。

相应的形状度量为

$$g=\frac{A}{\overline{d}^2}=\frac{N^3}{\left(\sum_{i=0}^{N}x_i\right)^2} \tag{10-43}$$

5．球状性

球状性（Sphericity）S 既可以描述二维目标也可以描述三维目标，其定义为

$$S=\frac{r_i}{r_c} \tag{10-44}$$

在二维情况下，r_i 代表区域内切圆的半径，而 r_c 代表区域外接圆的半径，两个圆的圆心都在区域的重心上，如图 10-13 所示。

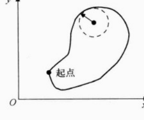


图 10-12　曲率半径

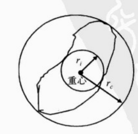


图 10-13　球状性定义示意图

当区域为圆时，球状性的值 S 达到最大值 1.0；而当区域为其他形状时，则有 $S<1.0$。S 不受区域平移、旋转和尺度变化的影响。

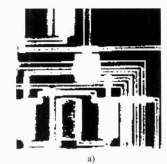
形状描述的 MATLAB 实现 1，程序代码如下：

```
%计算经过膨胀运算后图像面积的改变
BW=imread('circbw.tif');
subplot(1,2,1),imshow(BW);
%图像的膨胀
SE=ones(5);
BW1=imdilate(BW,SE);
subplot(1,2,2),imshow(BW1);
zengjia=(bwarea(BW1)-bwarea(BW))/bwarea(BW)
```

运行结果如下：

```
zengjia =
     0.3456
```

原始图像及膨胀后的图像如图 10-14 所示。


a)

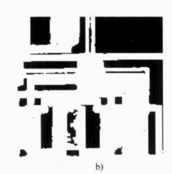

b)

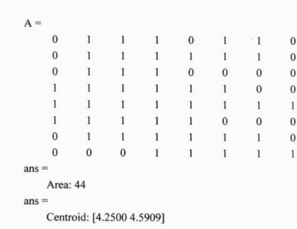
图 10-14　图像面积改变的应用
a) 原始图像　b) 膨胀后的图像

形状描述的 MATLAB 实现 2，程序代码如下：

```
%求矩阵 A 的面积和质心坐标
A=[0 1 1 1 0 1 1 0;
   0 1 1 1 1 1 1 0;
   0 1 1 1 0 0 0 0;
   1 1 1 1 1 1 0 0;
   1 1 1 1 1 1 1 1;
   1 1 1 1 1 0 0 0;
   0 1 1 1 1 1 1 0;
   0 0 0 1 1 1 1 1]
regionprops(A,'Area')
regionprops(A,'Centroid')
```



程序运行结果如下：



```
A =
     0     1     1     1     0     1     1     0
     0     1     1     1     1     1     1     0
     0     1     1     1     0     0     0     0
     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1
     1     1     1     1     1     0     0     0
     0     1     1     1     1     1     1     0
     0     0     0     1     1     1     1     1
ans =
    Area: 44
ans =
    Centroid: [4.2500 4.5909]
```



10.4 区域描述

对一幅灰度图像或者彩色图像运用图像分割的方法进行处理，把其中感兴趣的像素分离出来作为目标像素，取值为 1，而把不感兴趣的其余部分作为背景像素，取值为 0，就可以得到一幅二值图像。在理想情况下，希望该二值图像中的两个值准确地代表目标及背景。但实际中，往往所检测到的目标中还有若干个“假目标”出现，还有可能提取的是多个目标，因此，就需要对二值图像进行处理，实现对目标的分析。二值图像包含目标的位置、形状、结构等很多重要的信息，是图像分析和目标识别的依据。本节将围绕二值图像处理的方法进行阐述。

10.4.1 几何特征

1. 像素与领域

二值图像中的像素值不是 1，就是 0。其中 1 表示目标的值，0 表示背景的值。$f(x,y)$ 表示位于图像陈列中第 x 行、第 y 列的像素的值。一幅 $m\times n$ 的图像具有 m 行和 n 列，行的标号从 0 到 m–1，列的编号从 0 到 n –1。这样，$f(0,0)$ 表示图像左上角的像素值，$f(m-1,n-1)$ 表示图像右下角的像素值。

在许多算法中，当对某个像素进行运算时，不仅要用到该像素的值，也要用到它邻近像素的值。关于领域的定义，最常见的有两种，即 4-领域（4-neighbor）和 8-领域（8-neighbor）。图 10-15 为像素 (x, y) 的 4-领域和 8-领域示意图。

	$(x-1,y)$	
$(x,y-1)$	(x,y)	$(x,y+1)$
$(x+1,y)$		

a)

$(x-1,y-1)$	$(x-1,y)$	$(x-1,y+1)$
$(x,y-1)$	(x,y)	$(x,y+1)$
$(x+1,y)$	$(x+1,y)$	$(x+1,y+1)$

b)

图 10-15 像素(x,y)的 4-领域和 8-领域示意图

a) 4-领域 b) 8-领域

2. 区域面积

二值图像中目标物的面积 A 就是目标物所占像素点的数目，即区域的边界内包含的像素点数。目标物面积 A 的计算公式如下：

$$A=\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}f(x,y) \qquad (10\text{-}45)$$

对二值图像而言，若用 1 表示目标，用 0 表示背景，其面积就是统计 $f(x,y)=1$ 的个数。

3. 位置

由于目标在图像中总有一定的面积大小，因此有必要定义目标在图像中的精确位置。目标的位置有形心、质心之分，形心为目标形状的中心，质心为目标质量的中心。

对 $m\times n$ 大小的目标，其灰度值为 $f(x,y)$，质心 $(\overline{X},\overline{Y})$ 为

$$\overline{X}=\frac{1}{mn}\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}xf(x,y),\overline{Y}=\frac{1}{mn}\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}yf(x,y) \qquad (10\text{-}46)$$

形心为

$$\overline{x}=\frac{1}{mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}x_i,\overline{y}=\frac{1}{mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}y_j \qquad (10\text{-}47)$$

4. 区域周长

数字图像子集 S 周长的定义有不同概念，通常用下面三种定义来近似：

① 若将图像中每个像素都看做是单位面积的小方格，则区域和背景都由方格组成，区域的周长可以定义成区域和背景交界线（接缝）的长度。

② 将像素看做一个个的点，则区域周长可以定义为区域边界 8 方向链码的长度。

③ 区域周长用边界所占面积表示，也即边界点数和。

5. 方向

计算物体的方向比计算它的位置稍微复杂一点。某些形状（如圆）的方向不是唯一的，为了定义唯一的方向，一般假定物体是长形的，其长轴方向被定义为物体的方向。

图像中物体的二阶矩轴是这样一条线：物体上的全部点到该线的距离的平方和最小。给出一幅二值图像 $f(x,y)$，计算物体点到直线的最小二乘方拟合，使所有物体点到直线的距离的平方和最小，即

$$x^2=\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}r_{xy}^2f(x,y) \qquad (10\text{-}48)$$

式中，r_{xy} 是物体点 (x,y) 到直线的距离。

6. 距离

图像中两点 $P(x,y)$ 和 $Q(u,v)$ 之间的距离是重要的几何特性，常用以下 3 种方法测量：

1）欧几里德距离（Euclidean）。

$$d_e(P,Q)=\sqrt{(x-u)^2+(y-v)^2} \qquad (10\text{-}49)$$

2）4-领域距离（City-block 城区距离）。

$$d_4(P,Q)=|x-u|+|y-v| \qquad (10\text{-}50)$$

3）8-领域距离（Chessboard 棋盘距离）。

$$d_8(P,Q)=\max(|x-u|+|y-v|) \tag{10-51}$$

10.4.2 不变矩

由于图像区域的某些矩对于平移、旋转、尺度等几何变换具有一些不变的特性，因此，矩的表示方法在物体分类与识别方面具有重要的意义。

1. 矩的定义

对于二维连续函数 $f(x,y)$，$(j+k)$ 阶矩定义为

$$m_{jk}=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}x^j y^k f(x,y)\mathrm{d}x\mathrm{d}y \qquad j,k=0,1,2,\cdots,j,k\text{为整数} \tag{10-52}$$

由于 j 和 k 可取所有的非负整数值，因此，形成了一个矩的无限集。而且，这个集合完全可以确定函数 $f(x,y)$ 本身。也就是说集合 $\{m_{jk}\}$ 对于函数 $f(x,y)$ 是唯一的，也只有 $f(x,y)$ 才具有这种特定的矩集。

为了描述物体的形状，假设 $f(x,y)$ 的目标物体取值为 1，背景为 0，即函数只反映了物体的形状而忽略其内部的灰度级细节。

参数 $j+k$ 称为矩的阶。零阶矩是物体的面积，即

$$m_{00}=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(x,y)\mathrm{d}x\mathrm{d}y \tag{10-53}$$

当 $j=1$，$k=0$ 时，m_{10} 对二值图像来讲就是物体上所有点的 x 坐标的总和。同理，m_{01} 就是物体上所有点的 y 坐标的总和，令

$$\bar{x}=\frac{m_{10}}{m_{00}} \qquad \bar{y}=\frac{m_{01}}{m_{00}} \tag{10-54}$$

则 $(\bar{x},\bar{y})$ 就是二值图像中一个物体的质心的坐标。

中心矩定义为

$$\mu_{jk}=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(x-\bar{x})^j(y-\bar{y})^k f(x,y)\mathrm{d}x\mathrm{d}y \tag{10-55}$$

如果 $f(x,y)$ 是数字图像，由式（10-55）变为

$$\mu_{jk}=\sum_x\sum_y(x-\bar{x})^j(y-\bar{y})^k f(x,y) \tag{10-56}$$

2. 不变矩

定义归一化的中心矩为

$$\mu_{jk}=\frac{\mu_{jk}}{(\mu_{00})^{\gamma}}，\quad \gamma=\left(\frac{j+k}{2}+1\right) \tag{10-57}$$

利用归一化的中心矩，可以获得对平移、缩放、镜像和旋转都不敏感的 7 个不变矩，定义如下：

$$\varphi_1=\eta_{20}+\eta_{02} \tag{10-58}$$

$$\varphi_2=(\eta_{20}+\eta_{02})^2+4\eta_{11}^2 \tag{10-59}$$

$$\varphi_3=(\eta_{30}+3\eta_{12})^2+(3\eta_{21}-\eta_{03})^2 \tag{10-60}$$

$$\varphi_4=(\eta_{30}+\eta_{12})^2+(\eta_{21}+\eta_{03})^2 \tag{10-61}$$

$$\varphi_5=(\eta_{30}-3\eta_{12})(\eta_{21}+\eta_{03})[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2]+$$

$$(3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2] \quad (10\text{-}62)$$

$$\varphi_6=(\eta_{20}-\eta_{02})[(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2]+4\eta_{11}^2(\eta_{30}+\eta_{12})(\eta_{21}+\eta_{03}) \quad (10\text{-}63)$$

$$\varphi_7=(3\eta_{12}-\eta_{30})(\eta_{30}+\eta_{12})[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2]+$$
$$(3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}-\eta_{03})^2] \quad (10\text{-}64)$$

下面为对一幅图像，分别对其进行逆时针旋转 5°、垂直镜像、尺度缩小为原始图像的一半，分别求出原始图像及变换后的各个图像的七阶矩，可以得出这 7 个矩的值对于旋转、镜像及尺度变换不敏感，程序代码如下：

```
clear all
I=imread('pout.tif');
I1=I;
imshow(I1      );
I2=imrotate(I,5,'bilinear');
figure,imshow(I2);
I3=fliplr(I);
figure,imshow(I3)
I4=imresize(I,0.5,'bilinear');
figure,imshow(I4);
display('原始图像');
qijieju(I1);
dispaly('旋转变化')
qijieju(I2)
dispaly('镜像变化')
qijieju(I3);
dispaly('尺度变化')
qijieju(I4)
%求七阶矩函数
function qijieju(I0)
A=double(I0);
[nc,nr]=size(A);
[x,y]=meshgrid(1:nr,1:nc);
x=x(:);
y=y(:);
A=A(:);
m00=sum(A);
if m00==0
    m00=eps;
end
m10=sum(x.*A);
m01=sum(y.*A);
xmean=m10/m00;
yean=m01/m00;
cm00=m00;
cm02=(sum((y-ymean).^2.*A))/(m00^2);
cm03=(sum((y-ymean).^3.*A))/(m00^2.5);
```



```
cm11=(sum((x-ymean).*(y-ymean).*A))/(m00^2);
cm12=(sum((x-ymean).*(y-ymean).^2.*A))/(m00^2.5);
cm20=(sum((x-xmean).^2.*A))/(m00^2);
cm21=(sum((x-xmean).^2.*(y-ymean).*A))/(m00^2.5);
cm30=(sum((x-xmean).^3.*A))/(m00^2.5);
ju(1)=cm20+cm02;
ju(2)=(cm20-cm02)^2+4*cm11^2;
ju(3)=(cm30-3*cm12)^2+(3*cm21-cm03)^2;
ju(4)=(cm30+cm12)^2(cm21+cm03)^2;
ju(5)=(cm30-3*cm12)*(cm30+cm12)*((cm30+cm12)^2-3*(cm21+cm03)^2)+(3*cm21-
cm03)*(cm21+cm03)*(3*(cm30+cm12)^2-(cm21+cm03)^2);
ju(6)=(cm20-cm02)*((cm30+cm12)^2-(cm21+cm03)^2)+4*cm11*(cm30+cm12)*(cm21+cm03);
ju(7)=(3*cm21-cm03)*(cm30+cm12)*((cm30+cm12)^2-3*(cm21+cm03)^2)+(3*cm12-
cm30)*(cm21+cm03)*(3*(cm30+cm12)^2-2(cm21+cm03)^2);
qijieju=abs(log(ju))
```

程序运行结果如下：

原始图像

qijieju = 6.5235   16.3199   25.7319   24.5010   50.4446   32.6666   49.8227

旋转变化

qijieju = 6.5235   16.3198   25.7314   24.5009   50.4437   32.6664   49.8224

镜像变化

qijieju = 6.5235   16.3199   25.7319   24.5010   50.4446   32.6666   49.8227

尺度变化

qijieju = 6.5239   16.3550   25.7595   24.4960   50.4555   32.6799   49.8278

原始图像及变化后的图像如图 10-16 所示。


a)


b)


c)


d)

图 10-16　不变矩应用举例

a) 原始图像　b) 逆时针旋转 5°后的图像　c) 垂直镜像后的图像　d) 尺度缩小的图像

10.5 形态分析

数学形态学（Mathematics Morphology）形成于 1964 年，法国巴黎矿业学院马瑟荣（G. Matheron）和他的学生赛拉（J. Serra）在从事铁矿核的定量岩石学分析中提出了该理论。数学形态学是在集合代数的基础上通过物体和结构元素相互作用的某些运算得到物体更本质的形态（Shape）的，是用集合论方法定量描述目标几何结构的学科，其基本思想和方法对图像处理的理论和技术产生了重大的影响，已成为数字图像处理的一个主要研究领域，在文字识别、显微图像分析、医学图像、工业检测、机器人视觉等方面都有很成功的应用。

用数学形态学处理二值图像时，要设计一种搜集图像信息的“探针”，称为结构元素。结构元素通常是一些小的简单集合，如图形、正方形等的集合。如图 10-17 所示，观察者在图像中不断地移动结构元素，看是否能将这个结构元素很好地填放在图像的内部，同时验证填放结构元素的方法是否有效，并对图像内适合放入结构元素的位置做标记，从而得到关于图像结构的信息。这些信息与结构元素的尺寸和形状都有关。构造不同的结构元素（如方形或圆形结构元素），便可完成不同的图像分析，从而得到不同的分析结果。

用形态学的方法处理和分析图像即是对物体或目标的形态分析，本节主要介绍二值形态分析方法中最基本的几种运算，即腐蚀、膨胀以及由它们组合得到的开闭运算和边界检测算法。

1. 腐蚀

将一个集合 A 平移距离 b，可以表示为 $A+b$，其定义为

$$A+b=\{a+b\mid a\in A\} \tag{10-65}$$

图 10-18 说明了集合平移的过程，从几何上看，$A+b$ 表示 A 沿矢量 b 平移了一段距离。探测的目的，就是要标记出图像内部那些可以将结构元素填入的（平移）位置。

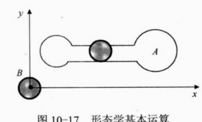


图 10-17　形态学基本运算

图 10-18　二值图像的平移

集合 A 被 B 腐蚀，表示为 $A\Theta B$，其定义为

$$A\Theta B=\{a:B+a\subset A\} \tag{10-66}$$

式中，A 为输入图像；B 为结构元素。

$A\Theta B$ 由将 B 平移 b 仍包含在 A 内的所有点 b 组成。如果将 B 看做模板，那么 $A\Theta B$ 则由在模板平移的过程中，所有可以填入 A 内部的模板的原点组成，如图 10-19 所示。

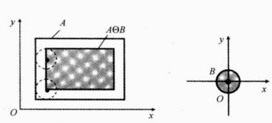


图 10-19 腐蚀类似于收缩

从图 10-19 中可以看出腐蚀是表示用某种“探针”（即结构元素）对一个图像进行探测，以便找出图像内部可以放下该结构元素的区域。它是一种消除边界点，使边界向内部收缩的过程。可以用来消除小且无意义的物体。一般而言，如果原点在结构元素的内部，则腐蚀后的图像为输入图像的子集，如果原点不在结构元素的内部，则腐蚀后的图像可能不在输入图像的内部，但输出形状不变，如图 10-20 所示。

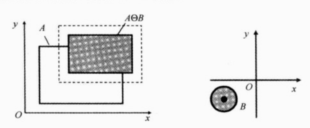


图 10-20 腐蚀不是输入图像的子图像

如用 0 代表背景，1 代表目标，设数字图像 S 和结构元素 E 为

$$\boldsymbol{S}=\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0_{\triangle} & 1 & 1 & 1 & 0 \end{pmatrix} \qquad \boldsymbol{E}=\begin{pmatrix} 1 & 0 \\ 1 & 1_{\triangle} \end{pmatrix}$$

三角形“△”代表坐标原点，则用 $\boldsymbol{E}$ 对 $\boldsymbol{S}$ 腐蚀的结果为

$$\boldsymbol{S}\Theta\boldsymbol{E}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0_{\triangle} & 0 & 1 & 1 & 0 \end{pmatrix}$$

2．膨胀

设有一幅图像 A，将 A 中所有元素相对原点转 180°，即令 (x_0,y_0) 变成 $(-x_0,-y_0)$，所得到的新集合称为 A 的对称集，记为 $-A$，如图 10-21 所示。

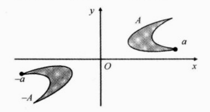


图 10-21 相对原点转 180°

以 A^C 表示集合 A 的补集，$-B$ 表示 B 关于坐标原点的反射（对称集）。那么，集合 A 被 B 膨胀，表示为 $A\oplus B$，其定义为

$$A\oplus B=[A^C\Theta(-B)]^C \tag{10-67}$$

为了利用结构元素 B 膨胀集合 A，可将 B 相对原点旋转 180°，得到$-B$，再利用$-B$ 对 A^C 进行腐蚀，腐蚀结果的补集就是所求的结果，如图 10-22 所示。

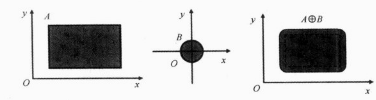
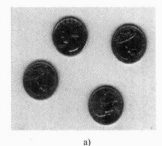


图 10-22　利用圆盘膨胀

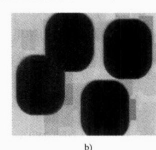
以下程序示例说明了如何对图像 eight 进行腐蚀和膨胀操作，程序代码如下：

```
%创建结构元素
SE=strel('rectangle',[40 30]);
I=imread('eight.tif');
figure(1),imshow(I);
%使用结构元素腐蚀图像
I2=imerode(I,SE);
figure(2),imshow(I2)
%恢复矩形为原有大小，使用相同的结构元素对腐蚀过的图像进行膨胀
I3=imdilate(I2,SE);
figure(3),imshow(I3)
```

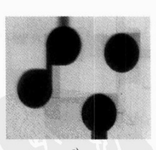
执行程序代码后，效果如图 10-23 所示。



a)



b)



c)

图 10-23　膨胀、腐蚀综合操作的显示效果

a) 原始图像　b) 腐蚀图像　c) 膨胀图像

膨胀的等效方程：膨胀还可以通过相对结构元素的所有点平移输入图像，然后计算并集得到，可用如下表达式描述：

$$A\oplus B=\bigcup\{A+b\mid b\in B\} \tag{10-68}$$

式（10-68）也称为明夫斯基和形式。图 10-24 是用式（10-68）膨胀的示意图，

图 10-24a 为输入图像，图 10-24b 为结构元素，将输入图像相对于结构元素内的 3 个点进行平移，并将 3 个平移图像叠加，最后的输出图像如图 10-24c 所示，图 10-24c 中 3 种不同外框标出的点对应了输入图像的 3 次平移。



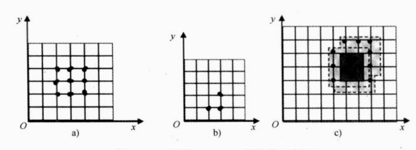


图 10-24　用式（10-68）膨胀的示意图

a) 输入图像　b) 结构元素　c) 膨胀结果

3. 开运算

假定 A 仍为输入图像，B 为结构元素，利用 B 对 A 作开运算，用 $A\circ B$ 表示，其定义为

$$A\circ B=(A\Theta B)\oplus B \tag{10-69}$$

所以，开运算实际上是 A 先被 B 腐蚀，然后再被 B 膨胀的结果。开运算通常用来消除小对象物、在纤细点处分离物体、平滑较大物体边界的同时并不明显改变其面积。图 10-25 是用圆盘对矩形进行开运算的例子。

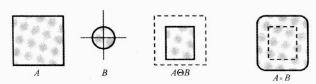


图 10-25　用圆盘对输入图像开运算的结果

从图 10-25 可以看到，开运算具有两个显著的作用：①利用圆盘可以磨光矩形内边缘，即可以使图像的尖角转化为背景。②用 $A-A\circ B$ 可以得到图像的尖角，因此圆盘的圆化作用可以起到低通滤波的作用。

开运算在粘连目标的分离及背景噪声（椒盐噪声）的去除方面有较好的效果。

4. 闭运算

闭运算是开运算的对偶运算，定义为先进行膨胀然后再进行腐蚀。利用 B 对 A 进行闭运算，用 $A\cdot B$ 表示，其定义为

$$A\cdot B=[A\oplus(-B)]\Theta(-B) \tag{10-70}$$

即用 $-B$ 对 A 进行膨胀，将其结果再用 $-B$ 进行腐蚀，闭运算通常用来填充目标内细小孔洞、连接断开的邻近目标、平滑其边界的同时并不明显改变其面积。图 10-26 描述了闭运算的过程及结果。显然，用闭运算对图形的外部进行滤波，仅仅磨光了凸向图像内部的边角。

闭运算在去除图像前景噪声（砂眼噪声）方面有较好的应用。

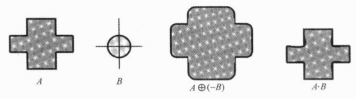


图 10-26　利用圆盘对输入图像进行闭运算

以下是利用 MATLAB 实现二值图像开运算和闭运算的程序代码：

```
clear all;
bw0=imread('F:\image\a.bmp')
figure(1),imshow(bw0)
%变为阈值取为 0.7 的二值图像
bw1=im2bw(bw0,0.7);
figure(2),imshow(bw1);
s=ones(3);
bw2=imopen(bw1,s);
figure(3),imshow(bw2);
bw3=imclose(bw1,s);
figure(4),imshow(bw3);
s1=strel('disk',2);
bw4=imopen(bw1,s1);
figure(5),imshow(bw4);
bw5=imclose(bw1,s1);
figure(6),imshow(bw5)
```

程序执行后，结果如图 10-27 所示。图 10-27c 是用三阶单位矩阵的结构元素进行腐蚀的结果，图 10-27d 是用三阶单位矩阵的结构元素进行膨胀的结果，图 10-27e 是用半径为 2 的平坦圆盘结构元素进行腐蚀的结果，图 10-27f 是用半径为 2 的平坦圆盘结构进行膨胀的结果。

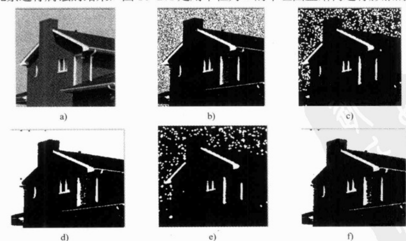
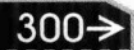


图 10-27　二值图像的腐蚀与膨胀

a) 原始图像　b) 二值化图像　c) 腐蚀后的图像 1　d) 膨胀后的图像 1　e) 腐蚀后的图像 2　f) 膨胀后的图像 2



5. 边界检测

利用圆盘结构元素进行膨胀会使图像扩大，进行腐蚀会使图像缩小，这两种运算都可以用来检测二值图像的边界。对于图像 A 和圆盘 B，图 10-28 给出了三种求取二值边界的方法：内边界、外边界和跨骑在实际边缘上的边界。其中跨骑在实际边缘上的边界又称为形态学梯度。图 10-29 是用腐蚀和膨胀方法得到的实际二值图像的边界实例。

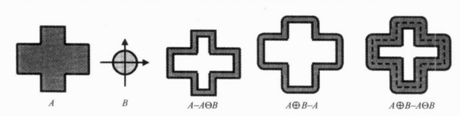
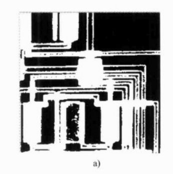
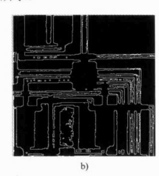


图 10-28 用腐蚀和膨胀运算得出的三种图像边界

下面为边界提取的具体代码，效果如图 10-29 所示。



a)



b)

图 10-29 边界提取示例

a) 原始图像 b) 边界提取效果

```
I=imread('circbw.tif');
imshow(I);
I2=bwperim(I,8); %获取区域的边界
figure,imshow(I2)
```

以上腐蚀、膨胀、开闭运算以及边界检测等都是二值形态分析处理方法中的基本运算，其他方法如细化、骨架算法等也都是比较常用的二值形态分析处理方法，而且这些算法还可以推广到一般的灰度图像处理上，限于篇幅，在此不再详细讨论，感兴趣的读者可以参阅相关资料。

10.6 区域、对象及特性度量

10.6.1 连通区域标记

在 MATLAB 7.0 图像处理工具箱中提供了专门的 bwlabel()函数，可以对二值图像的连接

部分进行标记。它主要对二值图像中各个分离部分进行标记。用户指定输入图像和特定的边缘约定类别，bwlabel()函数将返回与输入图像大小相同的数据矩阵，这样就可以利用数据矩阵各元素的不同整数值来区分图像中的不同物体。

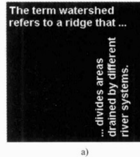
假设有二值图像如下：

```
BW =
     0     0     0     0     0     0     0     0
     0     1     1     0     0     1     1     1
     0     1     1     0     0     0     1     1
     0     1     1     0     0     0     0     0
     0     0     0     1     1     0     0     0
     0     0     0     1     1     0     0     0
     0     0     0     1     1     0     0     0
     0     0     0     0     0     0     0     0
调用 bwlabel()函数，同时指定 4-领域连接方式。
X=bwlabel(BW,4)
X =

     0     0     0     0     0     0     0     0
     0     1     1     0     0     3     3     3
     0     1     1     0     0     0     3     3
     0     1     1     0     0     0     0     0
     0     0     0     2     2     0     0     0
     0     0     0     2     2     0     0     0
     0     0     0     2     2     0     0     0
     0     0     0     0     0     0     0     0
```

在 X 中，1 代表第 1 个对象；2 代表第 2 个对象；3 代表第 3 个对象。如果采用 8-领域连接方式，则 4-领域中的对象 1 和对象 2 将合并为一个对象，这样就只能得到两个对象。

由于 bwlabel()函数的输出矩阵不是二值图像，而是 double 类型的矩阵，因此可以用索引色图像的格式显示该输出矩阵。在显示时，首先将各元素加 1，使各个像素值处于索引色图像有效的像素值范围中。这样，就可以将每个物体显示为不同的颜色，很容易将各个物体区分开来。例如，执行以下程序代码，显示结果如图 10-30 所示。



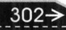
a)



b)

图 10-30 将图像中不同的物体表示为不同的颜色

a) 原始图像 b) 显示效果图像

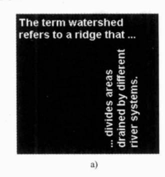


```
BW1=imread('text.png');
figure,imshow(BW1);
X=bwlabel(BW1,4);
RGB=label2rgb(X,@jet,'k');
figure,imshow(RGB+1,'notruesize')
```

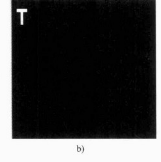
10.6.2 选择对象

在二值图像中，所谓对象就是指像素值为 1，且连接在一起的所有像素的集合。当只需要图像中特定的对象时，可使用对象操作选择想要的对象。在 MATLAB 7.0 图像处理工具箱中提供了专门的 bwselect()函数，用于在二值图像中选择特定的对象。在进行对象选择时，首先需要指定一些像素，然后 bwselect()函数才会返回包含指定像素的二值图像对象。

利用 bwselect()函数，可以实现某些特征提取，如提取文本图像中的某些字符对象。以下程序代码就是对如图 10-31a 所示的二进制文本图像提取其中的某些字符，其操作结果如图 10-31b 所示。



a)



b)

图 10-31 提取文本图像中字母的操作

a) 原始图像 b) 选择显示字母图像

```
BW1=imread('text.png');
figure,imshow(BW1);
c=[10 16 54];
r=[15 107 117]; %r、c 指定了图像中 T 字符
BW2=bwselect(BW1,c,r,4);
figure,imshow(BW2)
```

10.6.3 图像面积

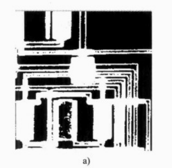
在进行图像处理时，有时会希望获取图像中改变某些特性的信息。例如，在进行膨胀操作时，我们希望知道有多少像素的值由 0 变为 1，或者想知道图像中对象的数目是否改变。对此可用 bwarea()函数和 bweuler()函数对图像特征进行提取操作，下面分别介绍。

在 MATLAB 7.0 图像处理工具箱中提供了专门的 bwarea()函数，可以用来获得二值图像的面积。这里的面积可以简单理解为图像前景中像素值为 1 的像素数量。

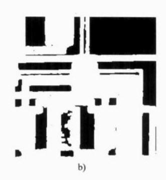


当然，bwarea()函数并非简单地计算像素值为 1 的像数数目，它还对不同的像素赋予不同的权值，以补偿由于用离散图像表示连续图像所引起的误差。例如，一条 50 点长的对角线要比 50 点长的水平线长。正是这个原因，bwarea()函数对 50 点长的水平线返回的面积为 50，而对 50 点长的对角线返回的面积则是 62.5。

以下程序代码示例将计算电路图（circbw.tif）经过膨胀运算后图像面积的改变量，如图 10-32 所示，图 10-32a 为膨胀前的图像，图 10-32b 为膨胀后的图像，经过计算，可知面积增加了 0.3456。



a)



b)

图 10-32　二值图像膨胀运算结果

a) 膨胀前的图像　b) 膨胀后的图像

```
BW=imread('circbw.tif');
figure,imshow(BW);
figure;
se=ones(5);
BW2=imdilate(BW,se);
imshow(BW2);
imcrease=(bwarea(BW2)-bwarea(BW))/bwarea(BW);
```

程序运算结果：

```
imcrease =
    0.3456
```

10.6.4 欧拉数

在对图像拓扑进行估计时，经常会使用到一个量——欧拉数。所谓欧拉数，就是一幅图像中的对象数目减去该图像中的孔洞数目。在模式识别中，经常利用欧拉数进行聚类分析，该分析方法是一种非常有效的方法。

在 MATLAB 7.0 图像处理工具箱中提供了专门的 bweuler()函数，用于计算二进制图像的欧拉数。

利用 bweuler()函数进行欧拉数计算时，该函数只支持 4-领域和 8-领域两种连通方式。如何利用 bweuler()函数进行欧拉数计算呢？以下程序代码示例说了该函数的使用情况，并在 8-领域方式下，计算了 trees 图像（见图 10-33）的欧拉数。

```
clear all;
BW=imread(' F:\image\trees.tif');
imshow(BW)
s=bweuler(BW,8);
```

程序运算结果：

```
s =
   -100
```

在该示例中，欧拉数是个负数，表明孔洞数目大于对象数目。

10.6.5 基于分水岭的图像分割示例

在本节，将以一个具体示例说明如何利用 MATLAB 7.0 图像工具箱中提供的形态学处理函数，实现对一幅图像进行有效分割。图像分割后，该图像中相连的多个对象将被分割成多个单对象。为此，分下面几步完成。

1. 读取图像

从图像处理工具箱中读取图像'ff.jpg'，如图 10-34 所示。

```
I=imread('F:\image\ff.jpg')
figure,
imshow(I);
```



图 10-33 trees 图像

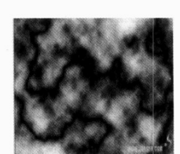


图 10-34 原始图像

从图 10-34 可知，该图像包含众多不同大小的、紧密相连的对象。为了有效分割开这些对象，可用分水岭变换方法进行处理。这是因为可以将图像看成一个凹凸表面，利用分水岭变换，可以有效发现图像中的谷点。

为了得到最好的结果，可以增强图像中感兴趣的对象来最小化，通过分水岭变换图像中谷点的个数。而对于对比度增强的一般技术是联合使用 top_hat 变换和 bottom_hat 变换。

2. 创建结构元素

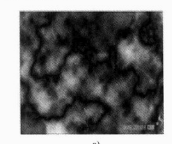
由于在图像中所感兴趣的对象像一个磁盘，所以该示例中可以用 strel()函数创建一个磁盘结构元素，磁盘的大小可用图像中对象的平均半径来估计。

```
Se=strel('disk',15);
```

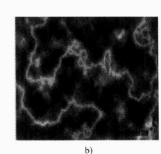


3. 增强图像对比度

Imtophat()函数和 imbothat()函数可以分别返回原图像的 top-hat 变换和 bottom-hat 变换，变换效果如图 10-35 所示。



a)



b)

图 10-35 增强图像对比度

a) top-hat 变换后的效果 b) bottom-hat 变换后效果

```
I=imread('F:\image\ff.jpg')
se=strel('disk',15);
top=imtophat(I,se);
bot=imbothat(I,se);
figure,imshow(top,[]);
figure,imshow(bot,[]);
```

4. 增大对象间的间隙

由图 10-36 可知，top-hat 图像包含了匹配结构元素的对象，但对象间比较紧密，间隙较小，需要增大对象间的间隙。MATLAB 7.0 提供了 imbothat()函数，可以有效反映对象间的间隙。为了增加图像中对象和间隙的对比度，在该示例中，首先将 top-hat 变换所得到的图像与原始图像相加，然后再减去 bottom-hat 变换所得到的图像，就可以有效地增加对象和间隙的对比度。以上操作由以下程序代码实现，处理效果如图 10-36 所示。

```
Ienhance=imsubtract(imadd(top,I),bot);
figure,imshow(Ienhance)
```

5. 感兴趣对象的转换

由于分水岭变换可以有效检测图像中的谷点，所以可以首先用 imcomplement()函数增强照亮图像中的谷点，以利于有效检测。谷点增强照亮效果如图 10-37 所示。

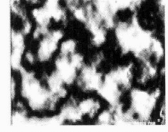


图 10-36 对象间的间隙增大效果图

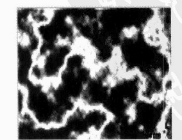
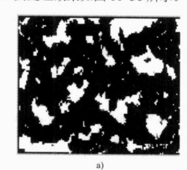


图 10-37 转化感兴趣的区域效果图

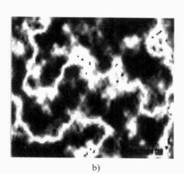
```
Iec=imcomplement(Ienhance);
figure,imshow(Iec)
```

6. 检测谷点

在增强照亮谷点之后，就可以有效地进行检测了。本例中可以使用 imextendedmin()函数和 imimposemin()函数来检测谷点，这两个函数的具体说明可参考图像处理工具箱，其引用方法如下，其处理效果如图 10-38 所示。



a)



b)

图 10-38　谷点检测效果图

a) imextendedmin 检测效果　b) imimposemin 检测效果

```
Iemin=imextendedmin(Iec,22);
Iemin=double(Iemin);  %转化为二进制图像
Iimpose=imimposemin(Iec,Iemin);
figure,imshow(Iemin);
figure,imshow(Iimpose)
```

7. 分水岭变换

通过上面的处理步骤，可以用 MATLAB 7.0 提供的 watershed()函数，使用以下程序代码对使用 imimposemin()函数进行谷点检测后的图像 Iimpose 进行分水岭变换操作。

Watershed()函数将返回一个包含非负元素的标签矩阵，该矩阵与分水岭区域相对应。不在分水岭区域的像素的像素值将全部设置为 0。

如果需要很好地可视化标签矩阵，则可以将标签矩阵转为颜色矩阵。利用 MATLAB 7.0 提供的 label2rgb()函数可以达到这一目的，可视化标签矩阵效果如图 10-39 所示。

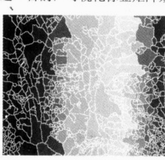


图 10-39　分水岭变换后的效果图

```
I=imread('trees.tif')
se=strel('disk',15);
top=imtophat(I,se);
bot=imbothat(I,se);
Ienhance=imsubtract(imadd(top,I),bot);
Iec=imcomplement(Ienhance);
Iemin=imextendedmin(Iec,22);
Iemin=double(Iemin);
Iimpose=imimposemin(Iec,Iemin);
wat=watershed(Iimpose);
rgb=label2rgb(wat);
figure,imshow(rgb)
```

8. 从标签矩阵中提取特征

当需要提取图像中某些特定的特征时，可以直接从标签矩阵中进行提取。MATLAB 7.0 提供的 regionprops()函数可以从标签矩阵中进行提取特征。例如，本例中需要计算两个观测量（面积和方向），并视它们为彼此的函数，则特征提取效果如图 10-40 所示。

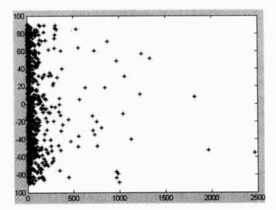


图 10-40 标签矩阵中提取特征效果图

执行程序代码如下：

```
stats=regionprops(wat,'Area','Orientation');
area=[stats.Area];
orient=[stats.Orientation];
figure,plot(area,orient,'b*');
```

习题

10-1 图像都有哪些特征？简要说明这些特征及其在图像分析中的用途。
10-2 数字图像处理技术中，计算区域面积有几种方法？
10-3 什么是纹理分析中的统计法？它有何特点？

10-4 对如图 10-41 所示的闭合边界，写出以 S 为起点的链码和差分链码。

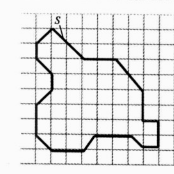


图 10-41

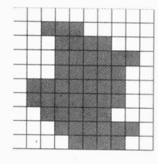


图 10-42

10-5 计算如图 10-42 所示图像中目标（阴影部分）的面积、质心、周长、圆形度、矩形度以及长宽比。

10-6 分别用方形和圆形两种结构元素实现二值图像的腐蚀运算，改变结构元素的大小，观察并分析腐蚀结果。

10-7 编程求二值图像的形态边界。



第 11 章　MATLAB 图像处理的应用

11.1　MATLAB 在遥感图像处理中的应用

遥感利用遥感器从空中来探测地面物体的性质，它根据不同物体对滤谱产生不同响应的原理，识别地面上各类地物，并经记录、传送、分析和判读来识别地物。本节主要介绍 MATLAB 遥感图像处理简介、遥感图像增强、图像融合、变换检测等几个方面的内容。通过本章的学习，我们应该对遥感有个大体的了解，掌握一些基本的遥感图像处理方法。

11.1.1　遥感简介

遥感作为一门对地观测综合性技术，它的出现和发展既是人们认识和探索自然界的客观需要，更有其他技术手段与之无法比拟的特点。从字面上说，遥感就是从远处感觉事物，严格的定义是远远地去感觉某一定对象的技术；而广义地讲，遥感是不直接接触地收集关于某一定对象的某种或某些特定的信息，从而了解这个对象的性质。遥感技术的特点归结起来主要有以下三个方面。

（1）探测范围广、采集数据快

遥感探测能在较短的时间内，从空中乃至宇宙空间对大范围地区进行观测，并从中获取有价值的遥感数据。这些数据拓展了人们的视觉空间，为宏观地掌握地面事物的现状创造了极为有利的条件，同时也为宏观地研究自然现象和规律提供了宝贵的第一手资料。这种先进的技术手段与传统的手工业相比是不可替代的。

（2）能动态反映地面事物的变化

遥感探测能周期性、重复地对同一地区进行对地观测，这有助于人们通过所获取的遥感数据，发现并动态地跟踪地球上许多事物的变化。同时，研究自然界的变化规律。尤其是在监视天气状况、自然灾害、环境污染甚至军事目标等方面，遥感的运用显得格外重要。

（3）获取的数据具有综合性

遥感探测所获取的是同一时段、覆盖大范围地区的遥感数据，这些数据综合地展现了地球上许多自然与人文现象，宏观地反映了地球上各种事物的形态与分布，真实地体现了地质、地貌、土壤、植被、水文、人工构筑物等地物的特征，全面地提示了地理事物之间的关联性，并且这些数据在时间上具有相同的现势性。

现在，世界各国都在利用陆地卫星所获取的图像进行资源调查（如森林调查、海洋泥沙和渔业调查、水资源调查等），灾害检测（如病虫害检测、水火检测、环境污染检测等），资源勘察（如石油勘查、矿产量探测、大型工程地理位置勘探分析等），农业规划（如土壤营养、水分和农作物生长、产量的估算等），城市规划（如地质结构、水源及环境分析等）。我国也陆续开展了以上各方面的一些实际应用，并获得了良好的效果。在气象预报和对太空其

他星球研究方面，数字图像处理技术也发挥了相当大的作用。

MATLAB 作为一个灵活实用的编程软件，早已渗透到遥感图像的处理中。利用 MATLAB 可以对遥感图像进行图像增强、滤波、灰度变换、图像融合、统计分析等，可以大大推动对遥感图像处理的深入研究和广泛应用。



MATLAB 在遥感图像中的应用主要包括以下几个方面。

1）军事侦察、定位、引导、指挥等。

2） 多光谱卫星图像分析。

3）地形、地图、国土普查。

4）地质、矿藏勘探。

5）森林资源探查、分类、防火。

6）水利资源探查，洪水泛滥检测。

7）海洋、渔业方面，如温度、鱼群的检测、预报。

8）农业方面，如谷物估产、病虫害调查。

9）自然灾害、环境污染的检测。

10）气象、天气预报图的合成分析预报。

11）天文、天空星体的探测及分析。

12）交通、空中管理、铁路选线等。

11.1.2 利用 MATLAB 对遥感图像进行直方图匹配

遥感系统记录地球表面物质的反射和发射辐射通量。理想情况下，某种物质的特定波长会反射大量的能量，而另一种物质在同样的波长下反射的能量可能要小得多。这使得遥感系统记录的两种地物之间存在对比度。然而，不同地物经常对可见光、近红外和中红外反射相似的辐射通量，使获取的影像对比度较低。另外，除了这些生物物理特征造成的明显低对比度外，人为因素也会对它产生影响。另一个导致遥感影像对比度低的因素是传感器的灵敏度。为了方便遥感影像分析人员对图像进行判读解译，需要对其进行增强处理。

在 MATLAB 中，使用 histeq()函数可以实现直方图匹配。

图 11-1a 显示了地球的一幅图像 f，图 11-1b 显示了使用 imhist(f)函数得到的直方图。由于这幅图像中存在大片的较暗区域，所以直方图中的大部分像素都集中在灰度级的暗端。乍一看，人们会认为利用直方图均衡化来增强图像是一种较好的方式，以便使较暗区域中的细节更加明显。然而，使用命令

```
g=histeq(f, 256)
```

得到如图 11-1c 所示的图像结果表明，利用直方图均衡化方法在此应用举例中并没有得到特别好的效果。对此，通过研究均衡化图像（见图 11-1d）可以看出原因。这里，我们看到灰度级已经移动到了灰度级的上半部分，因而输出图像出现了褪色现象。灰度级移动的原因是原始直方图中的暗色分量过于集中在 0 附近。从而，由该直方图得到的累计变换函数非常陡，因此才把在灰度级低端过于集中的像素映射到了灰度级的高端。直方图均衡化的程序代码如下：

```
%直方图均衡化
```

```
clear all
%%%显示原始图像
f=imread('moon.tif');
subplot(2,2,1),imshow(f); %原始图像
subplot(2,2,2),imhist(f); %原始图像的直方图
%%%对原始图像进行直方图均衡化
g=histeq(f,256);
subplot(2,2,3),imshow(g);
subplot(2,2,4),imhist(g);
```

执行程序代码后，效果如图 11-1 所示。

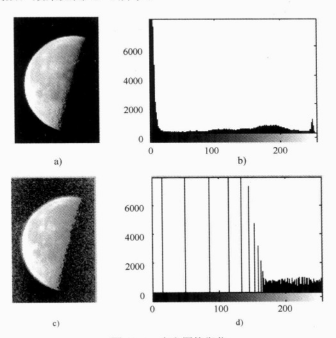


图 11-1　直方图均衡化

a) 原始图像　b) 原始图像的直方图　c) 直方图均衡化后的图像　d) 均衡化后的直方图

一种补偿这种现象的方法是使用直方图匹配，期望的直方图在灰度级低端应有较小的集中范围，并能够保留原始图像直方图的大体形状。由图 11-1b 可知，直方图主要有两个峰值，较大的峰值出现在原点处，较小的峰值出现在灰度级的高端。可使用多峰值高斯函数来模拟这种类型的直方图。

由于直方图均衡化在一些举例中出现的问题主要是原始图像 0 级的灰度附近像素过于集中，因而较为合理的手段是修改该图像的直方图，使其不再有此性质。图 11-2 显示了一个函数的图形（利用如下程序 manualhist 得到，参数分别为 0.15,0.05,0.75,0.05,1,0.07, 0.002），它不仅保留了原始直方图的大体形状，而且在图像的较暗区域中灰度级有较为平滑的过渡。

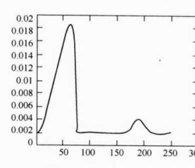


图 11-2 双峰高斯函数

```
function p=manualhist
repeats=true;
quitnow='x';
p=twomodegauss(0.15,0.05,0.75,0.05,1,0.07,0.002);
while repeats
    s=input('Enter m1,sig1,m2,sig2,A1,A2,k,OR x to quit:','s');
    if s==quitnow
        break
    end
    v=str2num(s);
    if numel(v)~=7;
        disp('Incorrect number of inputs')
        coninue;
    end
    p=twomodegauss(v(1),v(2),v(3),v(4),v(5),v(6),v(7));
    figure,plot(p);
    xlim([0 255]);
end
```

子函数 p=twomodegauss(m1,sig1,m2,sig2,A1,A2,k)计算一个已经归一化到单位区域的双峰高斯函数，以便可以将它用做一个指定的直方图。

```
function p=twomodegauss(m1,sig1,m2,sig2,A1,A2,k)
c1=A1*(1/((2*pi)^0.5)*sig1);
k1=2*(sig1^2);
c2=A2*(1/((2*pi)^0.5)*sig2);
k2=2*(sig2^2);
z=linspace(0,1,256);
p=k+c1*exp(-((z-m1).^2)./k1)+c2*exp(-((z-m2).^2)./k2);
p=p./sum(p(:));
```

程序的输出 p 由该函数产生的 256 个等间隔点组成，它是我们所希望的指定直方图。利用下面的命令可以得到具有指定直方图的图像，图 11-3 所示为最终结果。

```
gg=histeq(f,p)
```

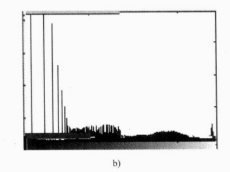
所用程序代码如下：

```
clear all;
%%%获取一个指定的函数
p=manualhist;
%%%使结果图像的直方图与获取函数图像一致
gg=histeq(f,p);
figure,
subplot(1,2,1),imshow(gg);
subplot(1,2,2);imhist(gg)
```

执行程序后，效果如图 11-3 所示。



a)



b)

图 11-3　直方图匹配增强

a) 直方图匹配增强的效果图　b) 期望图像的直方图

11.1.3 对遥感图像进行滤波增强

图 11-4a 所示为一幅月球图像。从图中可以看出，这幅图像比较模糊。在这种情况下，就需对图像进行增强处理（如图像锐化），同时尽可能地保留其灰度色调。首先，用下列语句生成一个拉普拉斯滤波器，其中心为–4。

```
w4=fspecial('laplacian', 0);
```

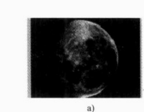
使用该滤波器对原始图像进行滤波，得到图 11-4b。可以看到，该图像比原始图像清晰了许多。为了得到更清晰的图像，也可以自己设计一个滤波器。本举例通过下面的命令设计一个中心为-32 的滤波器，得到图 11-4c。正如我们所预料的那样，它比其他两幅图像都要清晰。

```
w8=[1 1 1; 1 -32 1; 1 1 1];
```

本举例所用的程序代码如下，效果如图 11-4 所示。

```
clear all
f=imread('F:\image\moon.jpg');
subplot(1,3,1),imshow(f);
w4=fspecial('laplacian',0);
w8=[1 1 1;1 -32 1;1 1 1];
```

```
g4=f-imfilter(f,w4,'replicate');
g8=f-imfilter(f,w8,'replicate');
subplot(1,3,2);imshow(g4);
subplot(1,3,3);imshow(g8)
```


a)


b)


c)

图 11-4 图像滤波

a) 原始图像 b) 中心为-4 的滤波效果 c) 中心为-32 的滤波效果

11.1.4 对遥感图像进行融合

图像融合是一个对多遥感器的图像数据和其他信息的处理过程。它着重于把那些在空间和时空上冗余或互补的多源数据，按一定的规则（或算法）进行运算处理，获得比任何单一数据更精确、更丰富的信息，生成一幅具有新的空间、滤谱、时间特征的合成图像。它不仅是数据间的简单复合，还强调信息的优化，以突出有用的专题信息，消除或抑制无关的信息，改善目标识别的图像环境，从而增加解译的可靠性，减少模糊性（即多义性、不完全性、不确定性和误差），改善分类，扩大应用范围和效果。

1. 基于 HSI 彩色变换的图像融合

此举例是采用基于 HSI 彩色变换的小波融合算法。其基本思路是：将多光谱图像和高分辨率图像进行几何配准；然后对多光谱图像进行 HSI 变换，以提高多光谱彩色合成的解译能力；对 I（亮度）分量和高分辨率图像进行小波变换；然后保持多光谱图像亮度分量 I 的低频信息不变，将高分辨率图像小波分解后的高频信息叠加到多光谱图像亮度分量 I 的高频分量上，而后对同时具有低频信息和叠加后具有高频信息的亮度分量 I 进行小波逆变换，这样得出的 I 将会最大限度地保留原来多光谱图像的光谱信息，且能最大限度地提高其空间分辨率。最后将变换后的 H、S、I 分量在 RGB 三维空间进行级联，得到融合后的 RGB 空间图像。

需要注意的是，Chavez 等于 1991 年指出，用来进行多分辨率数据融合的所有方法中，HSI 方法造成光谱特征的畸变最严重，因此使用该方法时要谨慎，特别是需要对数据进行详细的辐射分析时。

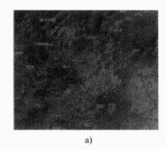
应用举例的程序代码如下：

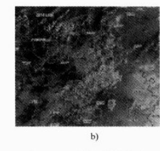
```
clear all;
%%%读取高分辨率图像
f1=imread('map1.jpg ');
subplot(2,2,1),imshow(f1)
%%%利用插值将多光谱图像放大到与高分辨率图像一样大小
[M,N]=size(f1);
f2=imread('map2.jpg ')
```



```
f2=imresize(f2,[M,N],'bilinear')
subplot(2,2,2);imshow(f2);
%%%将 RGB 空间转换为 HSI
f1=double(f1);
f2_hsi=rgb2hsv(f2);
f2_h=f2(:,:,1);
f2_s=f2(:,:,2);
f2_i =f2(:,:,3);
%%%进行小波分解
[c1 s1]=wavedec2(f1,1,'sym4');
f1=im2double(f1);
[c2_h s2_h]=wavedec2(f2_h,1,'sym4');
[c2_s s2_s]=wavedec2(f2_s,1,'sym4');
[c2_i s2_i]=wavedec2(f2_i,1,'sym4');
%%%对系数进行融合
c_h=0.5*(c2_h+c1);
c_s=0.5*(c2_s+c1);
c_i =c1;
%%%分别对 H 分量、S 分量、I 分量进行直方图均衡化
f_h= waverec2(c_h,s1,'sym4');
f_h=histeq(f_h);
f_s= waverec2(c_s,s1,'sym4');
f_s=histeq(f_s);
f_i = waverec2(c_i,s1,'sym4');
f_i=histeq(f_i);
%%%显示融合后图像
g=cat(3,f_h,f_s,f_i);
subplot(2,2,3);imshow(g);
```

执行程序后，结果如图 11-5 所示。


a)


b)

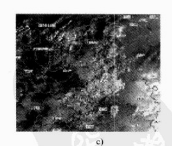

c)

图 11-5　遥感图像融合

a) 高分辨率图像　b) 多光谱图像　c) 融合后的图像

采用的数据源为 SPOT 10m 分辨率多光谱图像和 SPOT 2.5m 高分辨率图像如图 11-5a、b 所示。从图 11-5 上可以看出，SPOT 2.5m 高分辨率图像具有更多的道路网信息和许多大型建筑的边缘细节信息，而 SPOT 10m 分辨率多光谱图像则含有丰富的彩色信息，红色

表示植被覆盖（波段合成为 SPOT 近红外波段、红波段、绿波段），灰绿色为水体，褐色为建筑物。

图像融合的具体目标在于提高图像空间分辨率（图像锐化），改善图像几何精度，增强特征显示能力，改善分类精度，提供变换检测能力，替代或修补图像数据的缺陷等。经过融合，得到的结果图像如图 11-5c 所示。可以看到，融合后的图像不但有较好的空间特征和纹理特征，而且具有较好的多光谱保持能力。

2. 基于小波变换的图像融合

本书第 9 章介绍了小波变换图像融合的基本原理。本举例所用的高分辨率和多光谱图像如图 11-5a、b 所示，所用程序代码如下：

```
mul=imread('map.jpg');
hr=imread('map2.jpg')
[m,n]=size(hr);
mul=imresize(mul,[m,n],'bilinear');
mul_r=mul(:,:,1);
mul_g=mul(:,:,2);
mul_b=mul(:,:,3);
[c_hr s_hr]=wavedec2(hr,1,'sym4');
[c_r s_r]=wavedec2(mul_r,1,'sym4');
[c_g s_g]=wavedec2(mul_g,1,'sym4');
[c_b s_b]=wavedec2(mul_b,1,'sym4');
c_r=0.5*(c_hr+c_r);
c_g=0.5*(c_hr+c_g);
c_b=0.5*(c_hr+c_b);
f_r=waverec2(c_r,hr,'sym4');
f_r=histeq(f_r);
f_g=waverec2(c_g,hr,'sym4');
f_g=histeq(f_g);
f_b=waverec2(c_b,hr,'sym4');
f_b=histeq(f_b);
fc=cat(3,f_r,f_g,f_b);
figure;imshow(g);
```

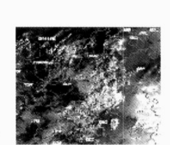


图 11-6　小波变换融合后的图像

执行程序后，效果如图 11-6 所示。

与传统的数据融合算法（如 HSI 等）相比，小波融合模型不仅能够针对输入图像的不同特征来合理选择小波基以及小波变换的次数，而且在融合操作时又可以根据实际需要来引入双方的细节信息，从而表现出更强的针对性和实用性，融合效果更好。另外，从实施过程的灵活性方面评价，HSI 变换只能而且必须同时对 3 个波段进行融合操作，PCA 分析的输入图像必须有 3 个或 3 个以上，而小波方法则能够完成对单一波段或多个波段进行的融合运算。

11.2　MATLAB 在医学图像处理中的应用

MATLAB 凭借其强大的矩阵运算功能和直观的编程风格。在医学图像处理中得到了广

泛的应用。本节主要介绍 MATLAB 医学图像处理概述、医学图像增强、灰度变换等内容。

11.2.1 医学成像简介

医学成像已经成为现代医疗不可或缺的一部分，其应用贯穿整个临床工作，不仅广泛用于疾病诊断，而且在外科手术和放射治疗等的计划设计、方案实施以及疗效评估方面发挥着重要的作用。目前，医学图像可以分为解剖图像和功能图像两个部分。解剖图像主要描述人体形态信息，包括 X 射线透射成像、CT、MRI、US，以及各类内窥（如腹腔镜及喉镜）获取的序列图像等。另外，还有一些衍生而来的特殊技术，如从 X 射线成像衍生而来的 DSA，从 MRI 技术衍生而来的 MRA，从 US 成像衍生而来的 Doppler 成像等。功能图像主要描述人体代谢信息，包括 PET、SPECT、fMRI 等。同时，也有一些广义的或者使用较少的功能成像方式，如 EEG、MEG、pMRI（perfusion MRI）、fCT 等。

在医学教学、科学研究以及临床工作中，我们要处理很多医学图像，借助 MATLAB 7.0 图像处理工具箱，可以大大提高工作效率。MATLAB 在医学图像处理中的应用主要包括以下方面。

1）显微图像处理。

2）DNA（脱氧核糖核酸）显示分析。

3）红、白血球分析计数。

4）虫卵及组织切片的分析。

5）癌细胞识别。

6）DSA（心血管数字减影）及其他减影技术。

7）内脏大小形态及异常检测。

8）微循环的分析判断。

9）心脏活动的动态分析。

10）热像分析，红外像分析。

11）X 光照片增强、冻结及伪色彩增强。

12）CT、MRI、γ 射线照相机，正电子和质子 CT 的应用。

13）生物进化的图像。

读者可以参阅相关专业书籍来查看这些方面的应用实例。这里从灰度变换、噪声去除等几个方面介绍 MATLAB 图像处理函数在医学图像处理中的应用。

11.2.2 医学图像的灰度变换

医学图像反映的是 X 射线穿透路径上人体各生理组织部位对 X 射线吸收量的累加值，而人体内生理组织是相互重叠的，一些组织结构由于与 X 射线吸收量较大的组织重叠而无法在 X 射线影像上清晰地显示。另外，在 CT 系统成像过程中，由于图像板中的磷粒子使 X 射线存在散射；在扫描过程中，激光扫描仪的激光在穿过图像板的深部时存在着散射，从而使图像模糊，降低了图像分辨率。应用图像增强处理方法凸显组织边缘和细节，成为医学图像处理的迫切需要。

图像增强就是一种基本的图像处理技术，增强的目的是对图像进行加工，以得到对医务工作者来说视觉效果更好、更易于诊断的图像。图像增强根据图像的模糊情况采用了各种特

殊的技术突出图像整体或局部特征，常用的图像增强技术有灰度变换、直方图处理、平滑滤波（高斯平滑）、中值滤波、梯度增强、拉普拉斯增强以及频域的高通、低通滤波等，这些算法运算量大、计算复杂、开始难度大。针对这些问题，可以在 MATLAB 环境中，利用 MATLAB 提供的功能强大的图像处理工具箱，简单快捷地得到统计数据，同时又可得到直观图示。



1. 使用 imshow()函数改变图像对比度

大家应该记得 imshow()函数，下面就回忆一下它的用法。该函数用于显示常规的图像，调用格式为

- imshow（f,G）

其中，f 是一个图像数组；G 是显示该图像的灰度级数。若将 G 省略，则默认的灰度级数是 256。该函数的另一种调用格式为

- imshow（f,[low,high]）

这种调用格式会将所有小于或等于 low 的值都显示为黑色，所有大于或等于 high 的值显示为白色。界于 low 和 high 之间的值将以默认的级数显示为中等亮度值。最后，调用格式

- imshow(f,[ ])

可以将变量 low 设置为数组 f 的最小值，将变量 high 设置为数组 f 的最大值。Imshow()函数的这一形式在显示一幅动态范围较小的图像或既有正值又有负值的图像时非常有用。在医学图像中，由于一系列原因，获取的信号会较弱，应用该函数对图像进行增强十分有效。

如图 11-7 所示，可以很明显地看到，动态范围很小，图像很暗，对比度很低，没有明显的亮区。这样的图像很难用眼睛去观察它内部包含的信息。使用 imshow()函数拉伸后，图像视觉效果明显得到改善，动态范围扩大。如果将原始图像的亮度值扩展到显示设备的全部动态范围，效果就更佳。



a)



b)

图 11-7　医学图像增强应用举例

a) 原始图像　b) 增强后的图像

此举例是使用 imshow()函数实现图像增强，程序代码如下：

```
clc
clear all
%从路径盘中读入一幅原始图像
I=imread('F:\image\Xmap.jpg');
figure,imshow(I);
title('原始图像')
%%%将原始图像进行增强操作
figure,imshow(I,[ ])
title('增强后图像')
```

2. 使用 imadjust()函数调整图像亮度

imadjust()函数是对灰度图像进行亮度变换的基本 IPT 工具。它的调用格式为

- g=imadjust(f,[low_in high_in],[low_out high_out],gamma)

该函数将图像 f 中的亮度值映射到 g 中，即将 low_in～high_in 之间的的值映射到 low_out～high_out 之间。low_in 以下的值与 high_in 以上的值则被剪切掉了；换句话说，low_in 以上的值映射为 low_out，high_in 以下的值映射为 high_out。输入图像应该为 uint8，uint16 或 double 型图像，输出图像与输入图像有着相同的类。除图像 f 外，imadjust()函数的所有输入参数均指定在 0～1 之间，而不论图像 f 的类型。若 f 是 uint8 型图像，则 imadjust()函数将乘以 255 来确定应用中的实际值；若 f 是 uint16 型图像，则 imadjust()函数将乘以 65535。若[low_in high_in]或[low_out high_out]使用空矩阵（[ ]）会得到默认值[0 1]。若 high_out 小于 low_out，则输出亮度会反转。

参数 gamma 指定了曲线的形状，该曲线用来映射 f 的亮度值，以便生成图像 g。若 gamma 小于 1，映射被加权至更高（更亮）的输出值；若 gamma 大于 1，则映射被加权至更低（更暗）的输出值；若省略了函数的参量，则 gamma 默认为 1（线性映射）。

以下命令将 05～075 之间的灰度级扩展到 0～1（见图 11-8）。

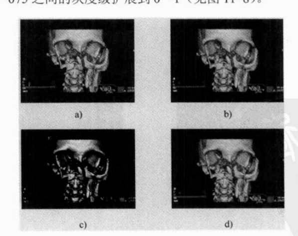


图 11-8　使用 imadjust()函数进行图像变换

a) 原始图像　b) 灰度反转后的图像　c) 部分区域灰度变换　d) gamma=2 的图像

```
%%%%%%
clc
clear
f=imread('F:\image\hand.jpg');
subplot(2,2,1);
imshow(f);
title('原始图像')
 %%%将原始图像灰度反转
g1=imadjust(f,[0 1],[0 1]);
subplot(2,2,2);
imshow(g1)
title('灰度反转后图像')
 %%%将原始图像 0.5～0.75 之间的灰度级扩展到[0 1]
g2=imadjust(f,[0.5 0.75],[0 1]);
subplot(2,2,3);
imshow(g2)
title('部分区域灰度变换')
 % 将 gamma 值设置为 2
g3=imadjust(f,[ ],[0 1]);
subplot(2,2,4);
imshow(g3)
title('gamma=2')
```

3. 自定义函数 intrans()

对图像的动态范围进行改变有很多方法，如对数、对比拉伸等。对数变换通过下式来实现：

$$g=c\log(1+\text{double}(f)) \tag{11-1}$$

式中，c 是一个常数。对数变换的一项主要应用是压缩动态范围。例如，傅里叶频谱的范围为[0 10^6]或更高。当傅里叶频谱显示于已线性缩放至 8bit 的监视器上时，高值部分占优，从而导致频谱中低亮度值的可视细节丢失。通过计算对数，10^6 左右的动态范围就会下降到可以接受、方便观察的范围，从而更有利于处理。

下面的函数称为对比度拉伸函数：

$$s=T(r)=\frac{1}{1+(m/r)^E} \tag{11-2}$$

式中，r 是输入图像的亮度；s 是输出图像中的相应亮度值；E 是该函数的斜率。由于 $T(r)$ 的限制值为 1，所以在执行此类变换时，输出值也被缩放在[0 1]内。因为该函数可以将输入值低于 m 的灰度级压缩在输出图像中较暗灰度级的较窄范围内；类似地，该函数可将输入值高于 m 的灰度级压缩在输出图像中较亮灰度级的较窄范围内。输出的是一幅具有较高对比度的图像。

根据以上所述，自定义 M 函数如下。

```
function g=intrans(f,varargin);
error(nargchk(2,4,nargin));
classin=class(f);
```

```
if strcmp(class(f),'double')&max(f(L))>1&~strcmp(varargin{1},'log')
    f=mat2gray(f);
else
    f=im2double(f);
end
method=varargin{1};
switch method
    case 'neg'
        g=imcomplement(f);
    case 'log'
        if length(varargin)==1
            c=1;
        elseif length(varargin)==2
            c=varargin{2};
        elseif length(varargin)==3
            c=varargin{2};
            classin=varargin{3};
        else
            eror('Incorrect number of inputs for    the log option')
        end
        g=c*(log(1+double(f)));
    case 'gamma'
        if length(varargin)<2
            error('Not enough inputs for the gamma option')
        end
        gam=varargin{2};
        g=imadjust(f,[],[],gam);
    case 'stretch'
        if length(varargin)==1
            m=mean2(f);
            E=4.0;
        elseif length(varargin)==3
            me=varargin{2};
            E=varargin{3};
        else error('Incorrect number of inputs for the tretch option')
        end
        g=1./(1+(m./(f+eps)).^E);
    otherwise
        error('UNknown enhancement method.')
end
```

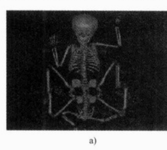
自定义 M 函数是将这些方法利用 MATLAB 来生成一个符合自己需要的函数，在这个函数中，用户可以根据自己的需要选择拉伸和拉伸因子。

图 11-9a 显示了一幅骨骼图像，通过图像可以观察到，原始图像对比度比较低，其中的大部分信息都不能用肉眼很好地观察到。使用自定义的 intrans()函数对其进行对比度拉伸，获得如图 11-9b 所示的结果。和图 11-9a 相比，图 11-9b 在视觉方面的改善效果是明显的。

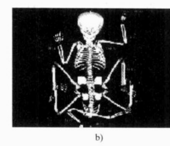
程序代码如下：

```
clear all;
f=imread('bone.jpg')
subplot(1,2,1);imshow(f);
%对图像进行拉伸
g=intrans(f,'stretch',mean2(im2double(f)),0.8);
subplot(1,2,2);imshow(g);
```

执行程序后，效果如图 11-9 所示。



a)



b)

图 11-9 使用自定义函数进行图像拉伸

a) 原始图像 b) 拉伸后的图像

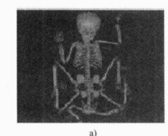
11.2.3 基于高频强调滤波和直方图均衡化的医学图像增强

高通滤波器削弱傅里叶变换的低频而保持了高频相对不变点，这样会突出图像的边缘和细节，使图像边缘更加清晰。但由于高通滤波器偏离了直流项，从而把图像的平均值降低到了零。一种补偿方法是给高通滤波器加上一个偏移量。若偏移量与滤波器乘以一个大于 1 的常数结合起来，则这种方法就称为高频强调滤波，因为该常量乘数突出了高频部分。这个乘数同时增加了低频部分的幅度，但只要偏移量与乘数项比较小，低频增强的影响就弱于高频的影响。高频强调滤波器的传递函数为。

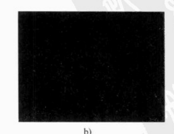
$$H_{\mathrm{hfe}}(u,v)=a+bH_{\mathrm{hp}}(u,v) \tag{11-3}$$

式中，a 是偏移量；b 是乘数；$H_{\mathrm{hp}}(u,v)$ 是高通滤波器的传递函数。

利用高频强调滤波和直方图均衡化对医学图像进行处理，如图 11-10 所示。

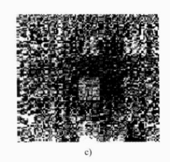


a)

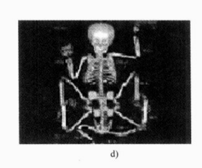


b)

图 11-10 高频强调滤波



c)



d)

图 11-10　高频强调滤波（续）

a) 原始图像　b) btw 滤波后的图像　c) 高频强调滤波后的图像　d) 直方图均衡化后的图像

```
%%%%%%
clc
clear
f=imread('lena.tif');
subplot(2,2,1); %在同一个窗口中显示多幅图像
imshow(f);
title('原始图像')
%%%对图像进行填充
PQ=paddedsize(size(f));
%%%高通滤波
D0=0.05*PQ(1);
HBW=hpfilter('btw',PQ(1),PQ(2),D0,2);
gbw=dftfilt(f,HBW);
gbw=uint8(gbw);
subplot(2,2,2);
imshow(gbw);
title('btw 滤波后图像')
%%%高频强调滤波
H=0.5+2*HBW;
ghf=dftfilt(f,H);
ghf=uint8(ghf);
subplot(2,2,3);
imshow(ghf)
title('高频强调滤波后图像')
%%%对高频强调滤波后，图像进行直方图均衡化
ghe=histeq(ghf,256);
ghe=uint8(ghe);
subplot(2,2,4);
imshow(ghe)
title('直方图均衡化图像')
```

上例所用子函数如下：

1）PQ=paddedsize（AB，CD，PARAM）对输入图像进行填充，以便形成的方形的大小

等于最近的 2 的整数次幂。

2）g=dftfilt(f,H)接收输入图像 f 和一个滤波函数 H，可处理所有的滤波细节并输出经滤波和剪切后的图像 g。

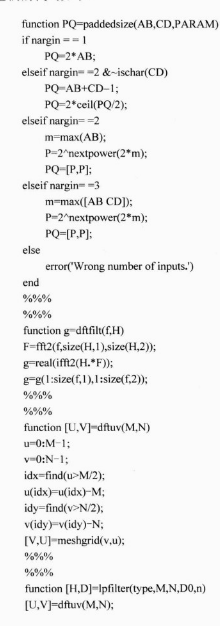


3）[U,V]=dftuv(M,N)提供了距离计算及其他类似应用所需要的网格数组。

4）[H,D]=lpfilter(type,M,N,D0,n)实现低通滤波。

5）H=hpfilter(type,M,N,D0,n)实现高通滤波。

它们的代码如下：



```
function PQ=paddedsize(AB,CD,PARAM)
if nargin = = 1
    PQ=2*AB;
elseif nargin= =2 &~ischar(CD)
    PQ=AB+CD-1;
    PQ=2*ceil(PQ/2);
elseif nargin= =2
    m=max(AB);
    P=2^nextpower(2*m);
    PQ=[P,P];
elseif nargin= =3
    m=max([AB CD]);
    P=2^nextpower(2*m);
    PQ=[P,P];
else
    error('Wrong number of inputs.')
end
%%%
%%%
function g=dftfilt(f,H)
F=fft2(f,size(H,1),size(H,2));
g=real(ifft2(H.*F));
g=g(1:size(f,1),1:size(f,2));
%%%
%%%
function [U,V]=dftuv(M,N)
u=0:M-1;
v=0:N-1;
idx=find(u>M/2);
u(idx)=u(idx)-M;
idy=find(v>N/2);
v(idy)=v(idy)-N;
[V,U]=meshgrid(v,u);
%%%
%%%
function [H,D]=lpfilter(type,M,N,D0,n)
[U,V]=dftuv(M,N);
```





```
D=sqrt(U.^2+V.^2);
switch type
    case 'ideal'
        H=double(D<=D0);
    case 'btw'
        if nargin= =4
            n=1;
        end
        H=1./(1+(D./D0).^(2*n));
    case 'gaussian'
        H=exp(-(D.^2)./(2*(D0^2)));
    otherwise
        error('UNknow filter type.')
end
%%%
%%%
function H=hpfilter(type,M,N,D0,n)
if nargin= =4
    n=1;
end
Hlp=lpfilter(type,M,N,D0,n);
H=1-Hlp;
```

习题

11-1　MATLAB 在遥感图像中的应用主要有哪几个方面？

11-2　小波变换图像融合与 HSI 图像融合相比具有哪些优势？试对同一幅图像同时进行 HIS 图像融合和小波变换图像融合。

11-3　MATLAB 在医学图像处理中的应用主要有哪几个方面？

11-4　试编写程序，自定义一个函数实现图像的位置及大小的改变。



附　　录

附录 A　MATLAB 6.X 图像处理工具箱函数

表 A-1　通用函数

函　数	功　能	语　法
colorbar	显示颜色条	colorbar colorbar('Vert') colorbar('horiz') colorbar(h) h= colorbar(…) colorbar(…,'peer',axes_handle)
getimage	从坐标轴取得图像数据	A=getimage(h) [x,y,A]= getimage(h) […,A,flag]= getimage(h) […]= getimage
imshow	显示图像	imshow(I,n) imshow(I,[low high]) imshow(BW) imshow(X,map) imshow(RGB) imshow(…,display_option) imshow(x,y,A,…) imshow filename h= imshow(…)
montage	在矩形框中同时显示多幅图像	montage(I) montage(BW) montage(X,map) montage(RGB) H= montage(…)
immovie	创建多帧索引色图像的电影动画	mov=immovie(X,map) mov= immovie(RGB)
subimage	在一幅图中显示多个图像	subimage(X,map) subimage(I) subimage(BW) subimage(RGB) subimage(x,y,…) h= subimage(…)
truesize	调整图像显示尺寸	truesize(fig ,[mrows mcols]) truesize(fig)
warp	将图像显示到纹理映射表面	warp(X,map) warp(I,n) warp(z,…) warp(x,y,z…) h= warp(…)
zoom	缩放图像	zoom on zoom iff zoom out zoom reset zoom zoom xon zoom yon zoom(factor) zoom(fig , option)

表 A-2 几何操作函数

函　数	功　能	语　法
imcrop	剪切图像	I2= imcrop(I) X2= imcrop(X,map) RGB2= imcrop(RGB) I2= imcrop(I,rect) RGB2= imcrop(RGB; rect) […]= imcrop(x,y,…) [A,rect]= imcrop(…) [x,y ,A,rect]= imcrop(…)
imresize	改变图像大小	B= imresize(A,m,method)
imrotate	旋转图像	B= imrotate (A,angel,method) B= imrotate(A,angel,method,'crop')

表 A-3 图像文件 I/O 函数

函　数	功　能	语　法
imfinfo	返回图形文件信息	info= imfinfo(filename, fmt) info= imfinfo(filename)
imread	从图形文件中读取图像	A= imread(filename,fmt) [X.,mpa]= imread(filename,fmt) […]= imread(filename) […]= imread(URL,…) […]= imread(…,idx) (CUR,ICO, and TIFF only) […]= imread(…, 'frames',idx) (GIF only) […]= imread(…,ref) (HDF only) […]= imread(…, 'BackgroundColor',BG) (PNG only) [A,map,alpha]= imread(…) (ICO,CUR, and PNG only)
imwrite	把图像写入图形文件中	imwrite(A,filename,fnt) imwrite(X,map,filename,fmt) imwrite(…,filename) imwrite(…,Param1,Val1,Param2,Val2,…)

表 A-4 线性滤波函数

函　数	功　能	语　法
conv2	进行二维卷积操作	C=conv2(A,B) C=conv2(hcol,hrow,A) C= conv2(…, 'shape')
convmtx2	计算二维卷积矩阵	T=convmtx2(H,m,n) T=convmtx2(H,[m n])
convn	计算 *n* 维卷积	C=convn(A,B) C=convn(A,B, 'shape')
filter2	进行二维线性过滤操作	Y= filter2(h,X) Y= filter2(h,X,shape)
fspecial	创建预定义过滤器	h= fspecial(type) h= fspecial(type,parameters)



表 A-5　像素和统计处理函数

函　数	功　能	语　法
corr2	计算两个矩阵的二维相关系数	r=corr2(A,B)
imcontour	创建图像数据的轮廓图	imcontour(I,n) imcontour(I,v) imcontour(x,y,…) imcontour(…,LineSpec) [C,h]= imcontour(…)
imfeature	计算图像区域的特征尺寸	stats= imfeature(L,measurements) stats= imfeature(L,measurements,n)
imhist	显示图像数据的柱状图	imhist(I,n) imhist(X,map) [count,X]= imhist(…)
impixel	确定像素颜色值	P= impixel(I) P= impixel(X,map) P= impixel(RGB) P= impixel(I,c,r) P= impixel(X,map,c,r) P= impixel(RGB,c,r) [c,r,P]= impixel(…) P= impixel(x,y,I,xi,yi) P= impixel(x,y,X,map,xi,yi) P= impixel(x,y,RGB,xi,yi) [xi,yi,P]= impixel(x,y,…)
improfile	沿线段计算剖面图的像素值	c= improfile c= improfile(n) c= improfile(I,xi,yi) c= improfile(I,xi,yi,n) [cx,cy,c]= improfile(…) [cx,cy,c,xi,yi]= improfile(…) […]= improfile(x,y,I,xi,yi) […]= improfile(x,y,I,xi,yi,n) […]= improfile(…,method)
mean2	计算矩阵元素的平均值	B=mean2(A)
pixval	显示图像像素信息	pixval on
std2	计算矩阵元素的标准偏移	B=std2(A)

表 A-6　图像增强函数

函　数	功　能	语　法
histeq	用柱状图均衡化增强对比	J= histeq(I,hgram) J= histeq(I,n) [J,T]= histeq(I,…) newmap= histeq(X,map,hgram) newmap= histeq(X,map) [newmap,T]= histeq(X,…)
imadjust	调整图像灰度值或颜色映射表	J= imadjust(I,[low_in high_in],[low_out high_out],gamma) newmap=imadjust(map,[low_in high_in],[low_out high_out],gamma) RGB2= imadjust(RGB1,…)
imnoise	增强图像的渲染效果	J= imnoise(I,type) J= imnoise(I,type,parameters)
medfilt2	进行二维中值过滤	B= medfilt2(A,[m,n]) B= medfilt2(A) B= medfilt2(A,'indexed',…)
ordfilt2	进行二维统计顺序过滤	B=ordfilt2(A,order,domain) B= ordfilt2(A,order,domain,S) B= ordfilt2(…,padopt)
wiener2	进行二维适应性去噪过滤处理	J= wiener2(I,[m n],noise) [J,noise]= wiener2(I,[m n])

表 A-7 图像分析函数

函　　数	功　　能	语　　法
edge	识别强度图像中的边界	BW= edge(I,'sobel') BW= edge(I, 'sobel',thresh) BW=edge(I, 'sobel',thresh,direction) [BW ,thresh]= edge(I,'sobel',…) BW= edge(I, 'prewitt') BW= edge(I, 'prewitt',thresh) BW= edge(I, 'prewitt',thresh, direction) [BW ,thresh]= edge(I, 'prewitt',…) BW=edge(I, 'roberts') BW=edge(I, 'roberts',thresh)
edge	识别强度图像中的边界	[BW ,thresh]=edge(I, 'roberts',…) BW= edge(I, 'log') BW= edge(I, 'log',thresh) BW= edge(I, 'log',thresh,sigma) [BW,threshold]=edge(I, 'log',…) BW= edge(I, 'zerocross',thresh,h) [BW,thresh]=edge(I, 'zerocross',…) BW= edge(I, 'canny') BW= edge(I, 'canny',thresh) BW= edge(I, 'canny',thresh,sigma) [BW,threshold]=edge(I, 'canny',…)
qtgetblk	获取四叉树中的块值	[vals,r,c]= qtgetblk(I,S,dim) [vals,idx]= qtgetblk(I,S,dim)
qtsetblk	设置四叉树中的块值	J= qtsetblk(I,S,dim,vals)

表 A-8 线性二维滤波设计函数

函　　数	功　　能	语　　法
freqspace	确定二维频率响应频率空间	[f1,f2]= freqspace(n) [f1,f2]= freqspace([m,n]) [x1,y1]= freqspace(…, 'meshgrid') f= freqspace(N) f= freqspace(N, 'whole')
freqz2	计算二维频率响应	[H,f1,f2]= freqz2(h,n1,n2) [H,f1,f2]= freqz2(h,[n2 n1]) [H,f1,f2]= freqz2(h,f1,f2) [H,f1,f2]= freqz2(h) […]= freqz2(h,…,[dx dy]) […]= freqz2(h,…,dx) freqz2(…)
fsamp2	用频率采样法设计二维 FIR 过滤器	h= fsamp2(Hd) h= fsamp2(f1,f2,Hd,[m n])
ftrans2	通过频率转换设计二维 FIR 过滤器	h= ftrans2(b,t) h= ftrans2(b)
fwind1	用一维窗口方法设计二维 FIR 过滤器	h= fwind1(Hd,win) h= fwind1 (Hd,win1,win2) h= fwind1 (f1,f2,Hd,…)
fwind2	用二维窗口方法设计二维 FIR 过滤器	h= fwind2(Hd,win) h= fwind2(f1,f2,Hd,win)

表 A-9 二进制图像操作函数

函 数	功 能	语 法
applylut	在二进制图像中利用 lookup 表进行边沿操作	A= applylut(BW,LUT)
bwarea	计算二进制图像对象的面积	total= bwarea(BW)
bweuler	计算二进制图像的欧拉数	eul= bweuler(BW,n)
bwfill	填充二进制图像的背景	BW2= bwfill(BW1,c,r,n) BW2= bwfill(BW1,n) [BW2,idx]=bwfill(…) BW2= bwfill(x,y,BW1,xi,yi,n) [x,y,BW2,idx,xi,yi]= bwfill(…) BW2= bwfill(BW1, 'holes',n) [BW2,idx]= bwfill(BW1, 'holes',n)
bwlabel	标注二进制图像中已连接的部分	L= bwlabel(BW,n) [L,num]= bwlabel(BW,n)
bwmorph	提取二进制图像的轮廓	BW2= bwmorph(BW1,operation) BW2= bwmorph(BW1,operation,n)
bwperim	计算二进制图像中对象的周长	BW2= bwperim(BW1) BW2= bwperim(BW1,CONN)
bwselect	在二进制图像中选择对象	BW2= bwselect(BW1,c,r,n) BW2= bwselect(BW1,n) [BW2,idx] =bwselect(…) BW2= bwselect(x,y,BW1,xi,yi) [x,y,BW2,idx,xi,yi]= bwselect(…)
dilate	放大二进制图像	BW2= dilate(BW1,SE) BW2= dilate(BW1,SE ,lag) BW2= dilate(BW1,SE,…,n)
erode	弱化二进制图像的边界	BW2= erode(BW1,SE) BW2= erode(BW1,SE,alg) BW2= erode(BW1,SE,…,n)
makelut	创建一个用于 applylut()函数的 look 表	lut= makelut(fun,n) lut= makelut(fun,n,P1,P2,…)

表 A-10 图像变换函数

函 数	功 能	语 法
dct2	进行二维离散余弦变换	B=dct2(A) B= dct2(A,m,n) B= dct2(A,[m n])
dctmtx	计算离散余弦变换矩阵	D= dctmtx(n)
fft2	进行二维快速傅里叶变换	Y= fft2(X) Y= fft2(X,m,n)
fftn	进行 *n* 维快速傅里叶变换	Y= fftn(X) Y= fftn(X,size)
fftshift(X)	把快速傅里叶变换的 DC 组件移到光谱中心	Y= fftshift(X) Y= fftshift(X,dim)
idct2	计算二维离散反余弦变换	B= idct2(A) B= idct2(A,m,n) B=idct2(A,[m n])
ifft2	计算二维快速傅里叶反变换	Y= ifft2(X) Y= ifft2(X,m,n)
ifftn	计算 *n* 维快速傅里叶反变换	Y= ifftn(X) Y= ifftn(X,siz)
iradon	进行反 radon 变换	I= iradon(P,theta) I= iradon(P,theta,interp,filter,d,n) [I,h]= iradon(…)
phantom	产生一个头部幻影图像	P= phantom(def,n) P= phantom(E,n) [P,E]= phantom(…)
radon	计算 radon 变换	R= radon(I,theta) [R,xp]= radon(…)

表 A-11　颜色空间转换函数

函　　数	功　　能	语　　法
hsv2rgb	转换 HSV 的值为 RGB 颜色空间	M= hsv2rgb(H)
rgb2hsv	转化 RGB 的值为 HSV 颜色空间	cmap= rgb2hsv(M)
rgb2ntsc	转化 RGB 的值为 NTSC 颜色空间	yiqmap= rgb2ntsc(rgbmap) YIQ= rgb2ntsc(RGB)
rgb2ycbcr	转化 RGB 的值为 YCBCR 颜色空间	ycbcrmap= rgb2ycbcr(rgbmap) YCBCR= rgb2ycbcr(RGB)
ycbcr2rgb	转化 YCBCR 的值为 RGB 颜色空间	rgbmap=ycbcr2rgb(ycbcrmap) RGB= ycbcr2rgb(YCBCR)
ntsc2rgb	转换 NTSC 的值为 RGB 颜色空间	rgbmap= ntsc2rgb(yiqmap) RGB= ntsc2rgb(YIQ)

表 A-12　边沿和块处理函数

函　　数	功　　能	语　　法
bestblk	确定进行块操作的块大小	siz= bestblk([m n],k) [mb,nb]= bestblk([m,n],k)
blkproc	实现图像的显示块操作	B= blkproc(A,[m n],fun) B= blkproc(A,[m n],fun,P1,P2,…) B=blkproc(A,[mn],[mborder nborder],fun,…)
col2im	将矩阵的列重新组织到块中	A= col2im(B,[m n],[mm nn],block_type) A= col2im(B,[m n],[mm nn])
colfilt	利用列相关函数进行边沿操作	B= colfilt(b,[m n],block_type,fun) B= colfilt(b,[m n],block_type,fun,P1,P2,…) B= colfilt(b,[m n], [mborder nborder],…) B= colfilt(A, 'indexed',…)
im2col	重调图像块为列	B= im2col(A,[m n],block_type) B= im2col(A,[m n]) B= im2col(A, 'indexed',…)
nlfilter	进行边沿操作	B= nlfilter(A,[m n],fun) B= nlfilter(A,[m n],fun,P1,P2,…) B= nlfilter(A, 'indexed',…)

表 A-13　区域处理函数

函　　数	功　　能	语　　法
roicolor	选择感兴趣的颜色区	BW= roicolor(A,low,high) BW= roicolor(A,v)
roifill	在图像的任意区域中进行平滑插补	J= roifill(I,c,r) J= roifill(I) J= roifill(I,BW) [J,BW]= roifill(…) J= roifill(x,y,I,xi,yi) [x,y,J,BW,xi,yi]= roifill(…)
roifilt2	过滤敏感区域	J= roifilt2(h,I,BW) J= roifilt2(I,BW,fun) J= roifilt2(I,BW,fun,P1,P2,…)
roipoly	选择一个敏感的多边形区域	BW= roipoly(I,c,r) BW= roipoly(I) BW= roipoly(x,y,I,xi,yi) [BW,xi,yi]= roipoly(…) [x,y,BW,xi,yi]= roipoly(…)

表 A-14 图像类型和类型转换函数

函 数	功 能	语 法
dither	通过抖动增加外观颜色分辨率，转换图像	X= dither(RGB,map) BW= dither(I)
gray2ind	转换灰度图像为索引色图像	[X,map]= gray2ind(I,n) [X,map]= gray2ind(BW,n)
grayslice	从灰度图像创建索引色图像	X= grayslice(I,n) X= grayslice(I,v)
im2bw	转换图像为二进制图像	BW= im2bw(I,level) BW= im2bw(X,map,level) BW= im2bw(RGB,level)
im2double	转换图像矩阵为双精度型	I2= im2double(I1) RGB2= im2double(RGB1) I= im2double(BW) X2= im2double(X1,'indexed')
double	转换数据为双精度型	double(X)
uint8	转换数据为 8 位无符号整型	i= uint8(x)
im2uint8	转换图像矩阵为 8 位无符号整型	I2= im2uint8(I1) RGB2= im2uint8(RGB1) I= im2uint8(BW) X2= im2uint8(X1,'indexed')
im2uint16	转换图像矩阵为 16 位无符号整型	I2= im2uint16(I1) RGB2= im2uint16(RGB1) I= im2uint16(BW) X2= im2uint16(X1, 'indexed')
uint16	转换数据为 16 位无符号整型	i=uint16(x)
ind2gray	把索引色图像转化为灰度图像	I= ind2gray(X,map)
ind2rgb	转化索引色图像为 RGB 真彩图像	RGB= ind2rgb(X,map)
isbw	判断是否为二进制图像	flag=isbw(A)
isgray	判断是否为灰度图像	flag= isgray(A)
isind	判断是否为索引色图像	flag= isind(A)
isrgb	判断是否为 RGB 真彩图像	flag= isrgb(A)
mat2gray	转化矩阵为灰度图像	I= mat2gray(A,[amin amax]) I= mat2gray(A)
rgb2gray	转换 RGB 图像或颜色映射表为灰度图像	I= rgb2gray(RGB) newmap= rgb2gray(map)
rgb2ind	转换 RGB 图像为索引色图像	[X,map]= rgb2ind(RGB,tol) [X,map]= rgb2ind(RGB,nl)
rgb2ind	转换 RGB 图像为索引色图像	X= rgb2ind(RGB,map) […]= rgb2ind(…,dither_option)

表 A-15 工具箱参数设置的函数

函 数	功 能	语 法
iptgetpref	获取图像处理工具箱参数设置	value= iptgetpref(prefname)
iptsetpref	设置图像处理工具箱参数	iptsetpref(prefname,value)

附录 B MATLAB 7.0 图像处理工具箱新增函数

表 B-1

函数	功能	语法
adapthisteq	限制对比度直方图均衡化	J= adapthisteq(I) J= adapthisteq(I,param1,val1,param2,val2,…)
applycform	用于色彩空间变换	out= applycform(I,C)
brighten	增加或降低颜色映射表的亮度	brighten(beta) newmap=brighten(beta) newmap=brighten(map,beta) brighten(fig,beta)
bwboundaries	描绘二进制图像边界	B= bwboundaries(BW) B= bwboundaries(BW,CONN) B= bwboundaries(BW,CONN,options) [B L]= bwboundaries(…) [B L N A]= bwboundaries()
bwtraceboundary	描述二进制图像中的物体	B= bwtraceboundary(BW,P,fstep) B= bwtraceboundary(BW,P,fstep,CONN) B= bwtraceboundary(…,N,dir)
cmpremute	MATLAB 高级应用——图像及影像处理 326	[Y,newmap]=cmpermute(X,map) [Y,newmap]=cmpermute(X,map,index)
cmunique	查找颜色映射表中特定的颜色及相应的图像	[Y,newmap]=cmunique(X,map) [Y,newmap]=cmunique(RGB) [Y,newmap]=cmunique(I)
decorrstretch	对多通道图像进行去相关处理	S= decorrstretch(I) S= decorrstretch(I,TOL)
dicomdict	获取或读取 DICOM 文件	dicomdict('set',dictionary) dictionary=dicomdict('get')
fan2para	把 fan-beam 投影转换为 para-beam 投影	P= fan2para(F,D) P= fan2para(…,param1,va;l1,param2,val2,…) [P,paralle1_1ocations,paralle1_rotation_angles]=fan2 para(…)
fanbeam	计算 Fan-Beam 变换	F= fanbeam(I,D) F= fanbeam(…,param1,val1,param2,val2,…) [F,sensor_positions,fan_rotation_angles]= fanbeam(…)
getline	用鼠标选择 ployline	[x,y]= getline(fig) [x,y]= getline(ax) [x,y]= getline [x,y]= getline(…, 'closed')
getpts	用鼠标选择像素点	[x,y]= getpts(fig) [x,y]= getpts(ax) [x,y]= getpts
getrect	用鼠标选择矩形	rect= getrect(fag) rect= getrect(ax) rect= getrect(fig)
iccread	读取 ICC 剖面	P= iccread(filename)
ifanbeam	计算逆 Fan-Beam 变换	I= ifanbeam(F,D) I= ifanbeam(…,param1,val1,param2,val2,…) [I,H]= ifanbeam(…)
im2java2d	将图像转换为 Java 缓冲图像	jimage= im2java2d(I) jimage= im2java2d(X,map)

（续）

函　数	功　能	语　法
imview	在图像阅览器中显示图像	imview(I) imview(RGB) imview(X,map) imview(I,range) imview(filename) imview(…, 'InitialMagnification',initial_mag) h= imview(…) imview close all
ippl	检查 IPPL 的存在	TF= ippl [TF B]= ippl
iptdemos	显示图像处理工具箱中的索引色图像	iptdemos
lab2double	把 L*a*b 数据转换为双精度	labd= lab2double(lab)
lab2uint16	把 L*a*b 数据转换为 16 位数据	lab16= lab2uint16(lab)
lab2uint8	把 L*a*b 数据转换为 8 位数据	lab8= lab2uint8(lab)
makecform	创建一个色彩转换结构	C= makecform(type) C= makecform(type,'whitepoint',WP) C= makecform('ice', src_profile, dest_profile) C= makecform('ice', src_profile, dest_profile, 'SourceRenderingIntent',src_intent, 'DestRenderingIntent', dest_intent) C= makecform('clut',profile,LUTtype) C= makecform('mattrc',MatTrc, 'Direction',direction)
para2fan	从 parral beam 数据计算 Fan-Beam 映射	F= para2fan(P,D) I= para2fan(…,param1,val1,param2,val2,…) [F,fan_positions,fan_rotation_angles]= fan2para(…)
ploy2mask	把多边形区域转换成 mask 区域	BW= ploy2mask(x,y,m,n)
sim2bw	转换图像为二进制图像	BW=im2bw(I,level) BW=im2bw(X,map,level) BW=im2bw(RGB,level)
uintlut	查找表中 A 的像素值	B= uintlut(A,LUT)
xyz2double	将颜色数据从 XYZ 转换到双精度	xyzd= xyz2double(XYZ)
xyz2uint16	将颜色数据从 XYZ 转换到十六进制	xyz16= xyz2uint16(xyz)





参 考 文 献

[1] Cai T, Brown L. Wavelet Shrinkage for Nonequispaced Samples[J]. Annals of Statistics, 2005, 26(5): 1783-1799.

[2] S Mallat. Multifrequency Channel Decomposition of Images and Wavelet Models[J]. IEEE Trans, 1998. ASSP-37(12): 2091-2110.

[3] S Mallat, S Zhong. Characterization of Signals from Multiscale Edges[J]. IEEE Trans, 2003, PAMI-14(7): 710-732.

[4] Donoho David L, Johnstone I M. Ideal Spatial Adaptation Via Wavelet Shrinkage[J]. Biometric,1998, 81: 425-455.

[5] Cohen A. Biorthogonal Wavelets[M]. In: C.K. Chui ed. Wavelets: A Tutorial in Theory and Application. Academic Press Ltd, 1999: 123-152.

[6] Sardy S, et al. Wavelet Shrinkage for Unequally Spaced data[J]. Statistics and Computing, 1999, 9(1): 65-75.

[7] Zhang Lei, Bao Paul, Wu Xiaolin. Hybrid Inter- and Intra-wavelet Scale Image Restoration[J]. Pattern Recognition Letter, 2003,36(8): 1737-1746.

[8] Cohen A. Biorthogonal Wavelets and Dual Filter[J]. In: M. Barland ed. Wavelets in Image Communication. Elsevier,1994: 1-26.

[9] Donoho David L. Denoising by Soft-thresholding[J]. IEEE Transactions on Information Theory,1999, 1(2): 115-122.

[10] S Mallat, W L Hwang. Singularity Detection and Processing with Wavelets[J]. IEEE Trans, 2002, IT-38(2): 617-643.

[11] Mathworks Inc. Wavelet Toolbox User's Guide[M]. 2004.

[12] 飞思科技术产品研发中心. MATLAB 6.5 辅助小波分析与应用[M]. 北京：电子工业出版社，2003.

[13] 飞思科技术产品研发中心. 小波分析理论与 MATLAB7 实现[M]. 北京：电子工业出版社，2005.

[14] 葛哲学，陈仲生. MATLAB 时频分析技术及其应用[M]. 北京：人民邮电出版社，2006.

[15] 杨福生. 小波变换的工程分析与应用[M]. 北京：科学出版社，1999.

[16] 薛毅. 数值分析与实验[M]. 北京：北京理工大学出版社，2005.

[17] 魏巍. 应用数学工具箱技术手册[M] 北京：国防工业出版社，2004.

[18] Wolfram S. The Mathematica Book[M]. Cambridge University Press，1988.

[19] 何东健. 数字图像处理[M]. 西安：西安电子科技大学出版社，2004.

[20] 阮秋琦. 实用数字图像处理[M]. 北京：电子工业出版社，2001.

[21] 崔屹. 数字图像处理技术与应用[M]. 北京：电子工业出版社，1997.

[22] 阮秋琦. 数字图像处理基础[M]. 北京：中国铁道出版社，1988.

[23] 刘榴娣. 实用数字图像处理[M]. 北京：北京理工大学出版社，2001.

[24] Rafael C.Gonzalez. Digital Image Proccessing Using MATLAB[M]. BeiJing: Publishing House of Electronics Industry，2003.





[25] 傅德胜，寿益禾. 图形图像处理学[M]. 南京：东南大学出版社，2002.
[26] 程正兴. 小波分析算法与应用[M]. 西安：西安交通大学出版社，1997.
[27] 陈武凡. 小波分析及其在图像处理中的应用[M]. 北京：科学出版社，2002.
[28] 陈传波. 数字图像处理[M]. 北京：机械工业出版社，2004.
[29] 余成波. 数字图像处理及 MATLAB 实现[M]. 重庆：重庆大学出版社，2003.
[30] 朱虹. 数字图像处理基础[M]. 北京：科学出版社，2005.
[31] 李朝晖. 数字图像处理及应用[M]. 北京：机械工业出版社，2004.
[32] 霍宏涛. 数字图像处理[M]. 北京：机械工业出版社，2003.
[33] 罗述谦，周果宏. 医学图像处理与分析[M]. 北京：科学出版社，2003.

